JRuby 10

Improve your Ruby with JIT, GC, and Libraries from the JVM

Hello!

- Charles Oliver Nutter
- @headius(@mastodon.social)
- headius@headius.com
- JRuby developer since 2004
- Full-time JVM language advocate


JRuby

What is JRuby?

- Ruby on the Java Virtual Machine
- Ruby implementation first, JVM language second
  - Many benefits from JVM ecosystem
- Ruby code should "just work"
  - Different extension API, no forking, parallel threads
- Thousands of **production** users, **18 years** of real-world use

What Is Important?

- **Usability**: compatibility, startup time, warmup time
- **Runtime features**: GC, JIT, monitoring, profiling, concurrency
- **Platform features**: mobile, server, desktop, integration, deployment
- **Performance**: straight line, scaling, parallelism, resource usage
- Different applications needs different capabilities

Getting Started

JRuby

The Ruby Programming Language on the JVM

Latest release - **9.3.4.0**

tar | zip | exe | exe(x64)

WHY JRUBY? GET STARTED LEARN GET INVOLVED

The Best of the JVM

- High performance
- Real threading
- Vast array of libraries

It's Just Ruby

Ruby 2.5 compatible

Platform Independent

- Easy to install
- Easy migration
- No hassles

JRuby Install

- Install a Java runtime
  - Java 21+ required for JRuby 10
  - Latest Java recommended (Java 24)
- Install JRuby
  - Recommended: Ruby installer, system package, Docker image
  - Download tarball/zip, Windows installer, build yourself

```
$ java -version
openjdk version "24" 2025-03-18
OpenJDK Runtime Environment (build 24+36-3646)
OpenJDK 64-Bit Server VM (build 24+36-3646, mixed mode, sharing)
$ ruby-install jruby-10.0.0.1
>>> Updating jruby versions ...
>>> Installing jruby 10.0.0.1 into /Users/headius/.rubies/jruby-10.0.0.1 ...
>>> Downloading https://repo1.maven.org/maven2/org/jruby/jruby-dist/10.0.0.1/jruby-
dist-10.0.0.1-bin.tar.gz into /Users/headius/src ...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 32.1M  100 32.1M    0     0  7887k      0  0:00:04  0:00:04 --:--:-- 7887k
>>> Verifying jruby-dist-10.0.0.1-bin.tar.gz ...
>>> Extracting jruby-dist-10.0.0.1-bin.tar.gz to /Users/headius/src/jruby-10.0.0.1 ...
>>> Installing jruby 10.0.0.1 ...
>>> Symlinking bin/ruby to bin/jruby ...
>>> Successfully installed jruby 10.0.0.1 into /Users/headius/.rubies/jruby-10.0.0.1
```

```
[] ~ $ irb
>> RUBY_VERSION
=> "3.4.2"
>> JRUBY_VERSION
=> "10.0.1.0-SNAPSHOT"
>> runtime = java.lang.Runtime.runtime
 => #<Java::JavaLang::Runtime:0x64a896b0>
>> runtime.available_processors
 => 8
>> runtime.free_memory
 => 91420584
>>
```

Ruby Compatibility

- **JRuby 10** leaps to **Ruby 3.4**
  - Language and core specs: **98% passing**
  - Pure-Ruby standard library shared with CRuby
  - JRuby support for most native stdlib
- **JRuby 9.4** supports **Ruby 3.1**
  - Maintained until April 2026

JVM Libraries

A Whole New World

- JVM ecosystem has tens of thousands of libraries
  - Graphics, GUIs, servers, document formats, AI/LLM
  - One of the largest collections in the dev world
- All available to JRuby users!
  - Easy integration into Ruby apps and code

Search Results for pdf

Sort by: Best Match

Showing 1139 results out of the 692409 available packages

pdf.js

org.webjars.npm

WebJar for pdf.js

#Build Tools #Packaging

Latest version: 0.1.0 View all
Published: 4 years ago
Licenses: MIT
Used in: 0 projects
OSS Index: No vulnerabilities fou... View

org.jfree.pdf

org.jfree

JFreePDF is an API that provides a Graphics2D implementation that generates PDF output.

#Graphics

Latest version: 2.0.1 View all
Published: 2 years ago
Licenses: GNU General Public License (...
Used in: 0 projects
OSS Index: No vulnerabilities fou... View

pdf.js-viewer

org.webjars.bower

WebJar for pdf.js-viewer

#Build Tools #Packaging

Latest version: 1.6.211 View all
Published: 7 years ago
Licenses: Apache-2.0
Used in: 1 project
OSS Index: No vulnerabilities fou... View

fluid-pdf-jvm

io.fluidsonic.pdf

Easy PDF generation with HTML & CSS using Chromium or Google Chrome

#Uncategorized

Latest version: 0.20.0 View all
Published: 1 year ago
Licenses: Apache License 2.0
Used in: 0 projects
OSS Index: No vulnerabilities fou... View

Feedback

Search Results for llm

Sort by: Best Match

Showing 794 results out of the 692410 available packages

llm

LLM Agent Builder

io.github.llmagentbuilder

Latest version: 0.4.3 View all
Published: 4 months ago
Licenses: The Apache Software License...
Used in: 0 projects
OSS Index: No vulnerabilities fou... View

llm

LLM Agent Builder

com.javaaidev.llmagentbuilder

Latest version: 0.5.0 View all
Published: 1 month ago
Licenses: The Apache Software License...
Used in: 0 projects
OSS Index: No vulnerabilities fou... View

llm-agent-vector-qdrant

Parent POM for llm-agent-vector submodules

fun.krinsiman.llm

Latest version: 1.0.0-beta.6 View all
Published: 29 days ago
Licenses: Apache License, Version 2.0
Used in: 0 projects
OSS Index: No vulnerabilities fou... View

llm-agent-embeddings

Parent POM for llm-agent-embedding submodules

fun.krinsiman.llm

Latest version: 1.0.0-beta.6 View all
Published: 29 days ago
Licenses: Apache License, Version 2.0
Used in: 0 projects
OSS Index: No vulnerabilities fou... View

search *for llm*

Advanced Search →

DISPLAYING GEMS **1 - 30** OF **75** IN TOTAL

FILTER: **NAME (22)** **DESCRIPTION (39)** **SUMMARY (43)** **UPDATED LAST WEEK (4)** **UPDATED LAST MONTH (12)**

llm_memory *0.1.14*

A Ruby Gem for LLMs like ChatGPT to have memory using in-context learning

16,334

DOWNLOADS

ruby_llm *1.2.0*

Beautiful Ruby interface to modern AI

26,133

DOWNLOADS

Example: JFreeChart

- Chart-generating library with support for document output
- Easily used from JRuby
- https://blog.headius.com/2025/05/3d-charts-and-more-with-jruby-and-jfreechart.html

Dependencies

- Jarfile, like Gemfile
  - Like Bundler's Gemfile
  - List of libraries and versions from Maven
- lock_jars command
  - Like bundle install

```
jar 'org.jfree:jfreechart:1.5.5'
jar 'org.jfree:org.jfree.chart3d:2.1.0'
```

```
$ lock_jars

-- jar root dependencies --

      org.jfree:jfreechart:1.5.5:compile
      org.jfree:org.jfree.chart3d:2.1.0:compile
      org.jfree:org.jfree.svg:5.0.6:compile
      org.jfree:org.jfree.pdf:2.0:compile

Jars.lock updated
```

Import or Set Constants

```
Color = java.awt.Color
Rectangle = java.awt.Rectangle
BufferedImage = java.awt.image.BufferedImage

java_import java.awt.Color
java_import java.awt.Rectangle
java_import java.awt.image.BufferedImage
java_import javax.imageio.ImageIO
java_import org.jfree.chart3d.Chart3DFactory
java_import org.jfree.chart3d.data.DefaultKeyedValues
...
```

Load Data into Dataset

```
require 'json'

dataset = StandardCategoryDataset3D.new
data = JSON.load(File.read("data/app_monitoring_revenue.json"))
data.each do |name, subset|
  values = DefaultKeyedValues.new
  subset.each { values.put(_1, _2) }
  dataset.add_series_as_row name, values
end
```

Create a 3D Bar Chart

```
chart = Chart3DFactory.create_bar_chart(
  "Quarterly Revenues",
  "Application & Performance Monitoring Companies",
  dataset, nil, "Quarter",
  "$million Revenues")
chart.chart_box_color = Color.new(255, 255, 255, 127)
chart.legend_anchor = LegendAnchor::BOTTOM_RIGHT
```

Adjust Rendering

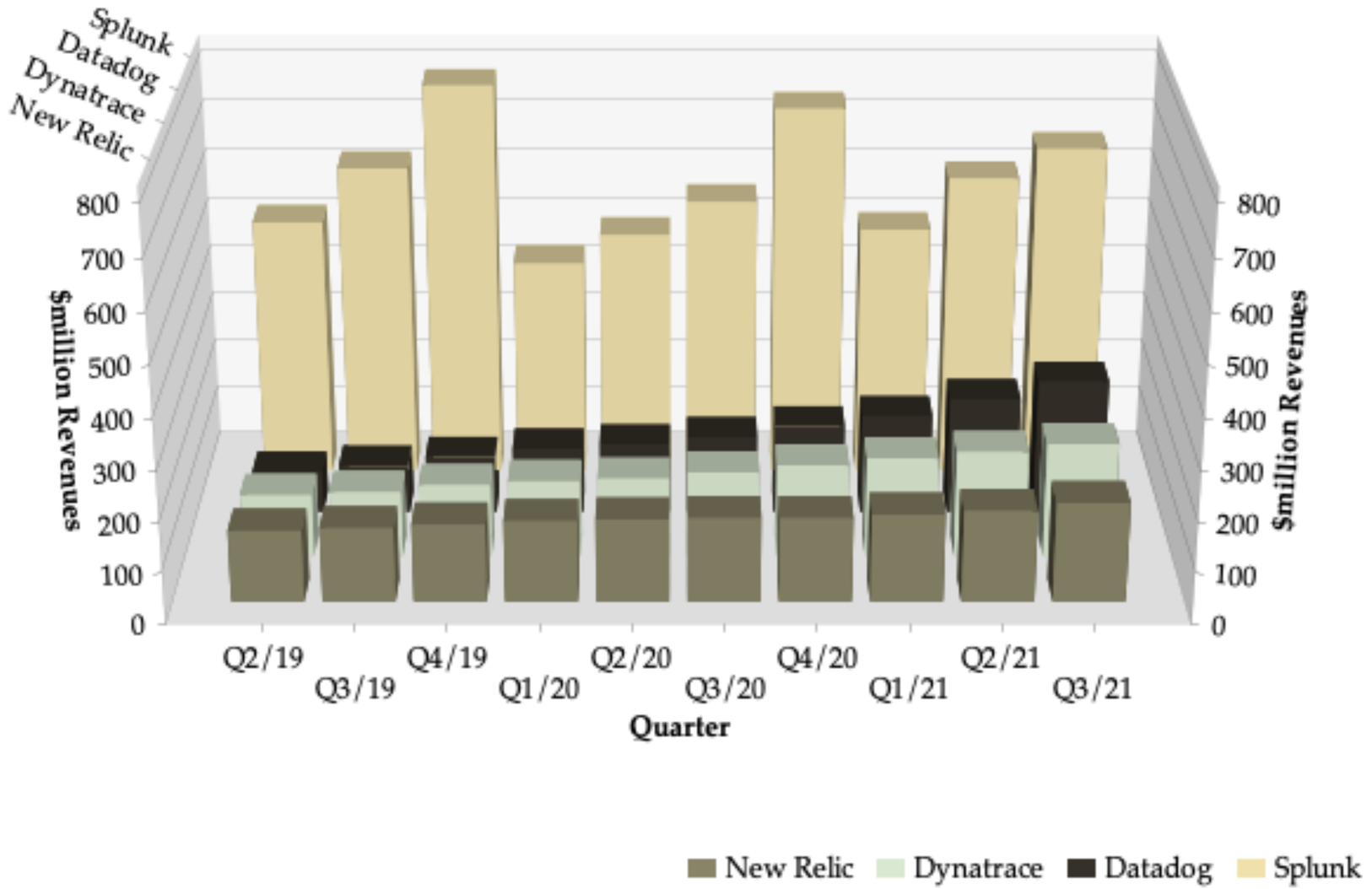
```
plot = chart.plot
plot.gridline_paint_for_values = Color::BLACK

renderer = plot.renderer
item_label_generator =
    StandardCategoryItemLabelGenerator.new(
      StandardCategoryItemLabelGenerator::VALUE_TEMPLATE)
item_selection = StandardKeyedValues3DItemSelection.new
item_label_generator.item_selection = item_selection
renderer.item_label_generator = item_label_generator
```

Render to PNG

```
width, height = 600, 500
category_chart_image =
    BufferedImage.new(width, height, BufferedImage::TYPE_INT_RGB)

category_chart_graphics =
    category_chart_image.create_graphics

chart.draw(category_chart_graphics, Rectangle.new(width, height))

category_chart_file =
    File.open("category_chart.png", "w")

ImageIO.write(category_chart_image, "PNG",
              category_chart_file.to_outputstream)
```

Quarterly Revenues

Application & Performance Monitoring Companies



Render to SVG and PDF

```
require_jar 'org.jfree', 'org.jfree.svg', '5.0.6'
java_import org.jfree.svg.SVGGraphics2D

svg_graphics = SVGGraphics2D.new(width, height)
svg_graphics.defs_key_prefix = "jruby_charts"
chart.element_hinting = true
chart.draw(svg_graphics, Rectangle.new(width, height))
svg = svg_graphics.get_svg_element chart.id
File.write("category_chart.svg", svg)
```

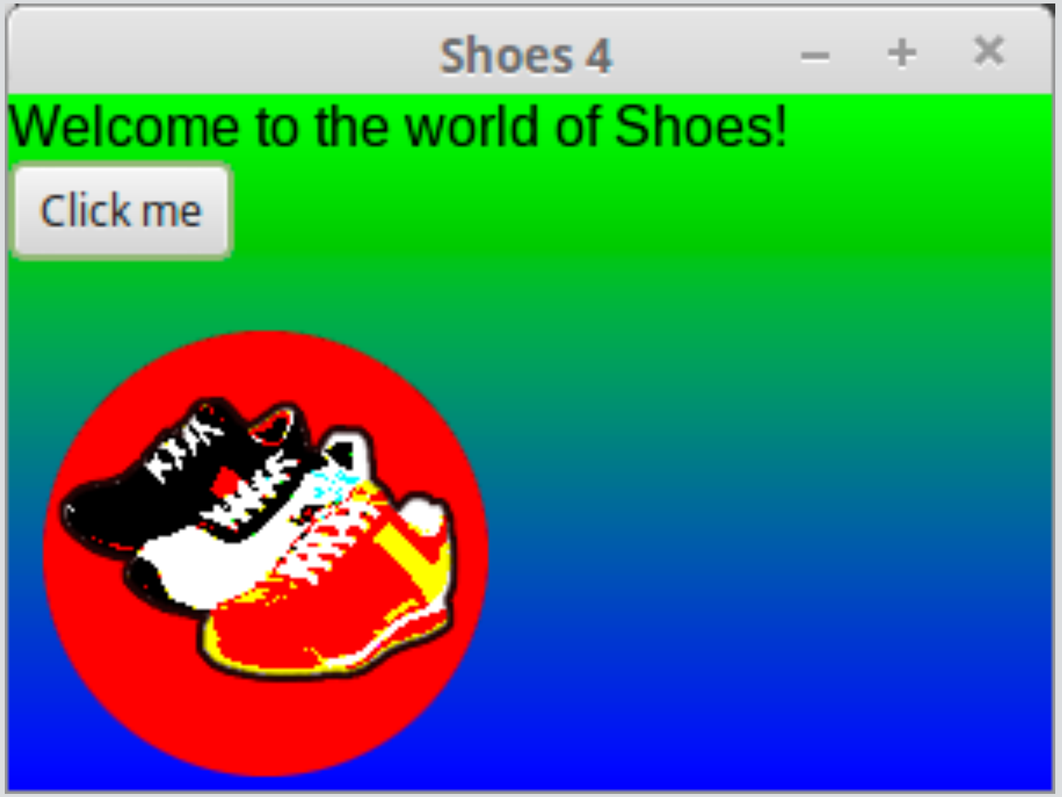
```
require_jar 'org.jfree', 'org.jfree.pdf', '2.0.1'
java_import org.jfree.pdf.PDFDocument

pdf_doc = PDFDocument.new
pdf_doc.title = "Application & Performance Monitoring Companies Revenue"
pdf_doc.author = "Charles Oliver Nutter";
page = pdf_doc.create_page(Rectangle.new(612, 468))
pdf_graphics = page.graphics2D
chart.draw(pdf_graphics, Rectangle.new(0, 0, 612, 468))
File.write("category_chart.pdf", pdf_doc.pdf_bytes)
```

GUI Libraries

- Swing, built into JDK
  - Clean, cross-platform, easy to build simple UIs
- Scalable Windowing Toolkit (Eclipse SWT)
  - Native widgets, WebKit browser component, rich ecosystem
- JavaFX (via JRubyFX, github/jruby/jrubyfx)
  - Scene-based, vector drawing, event-driven modern UI library

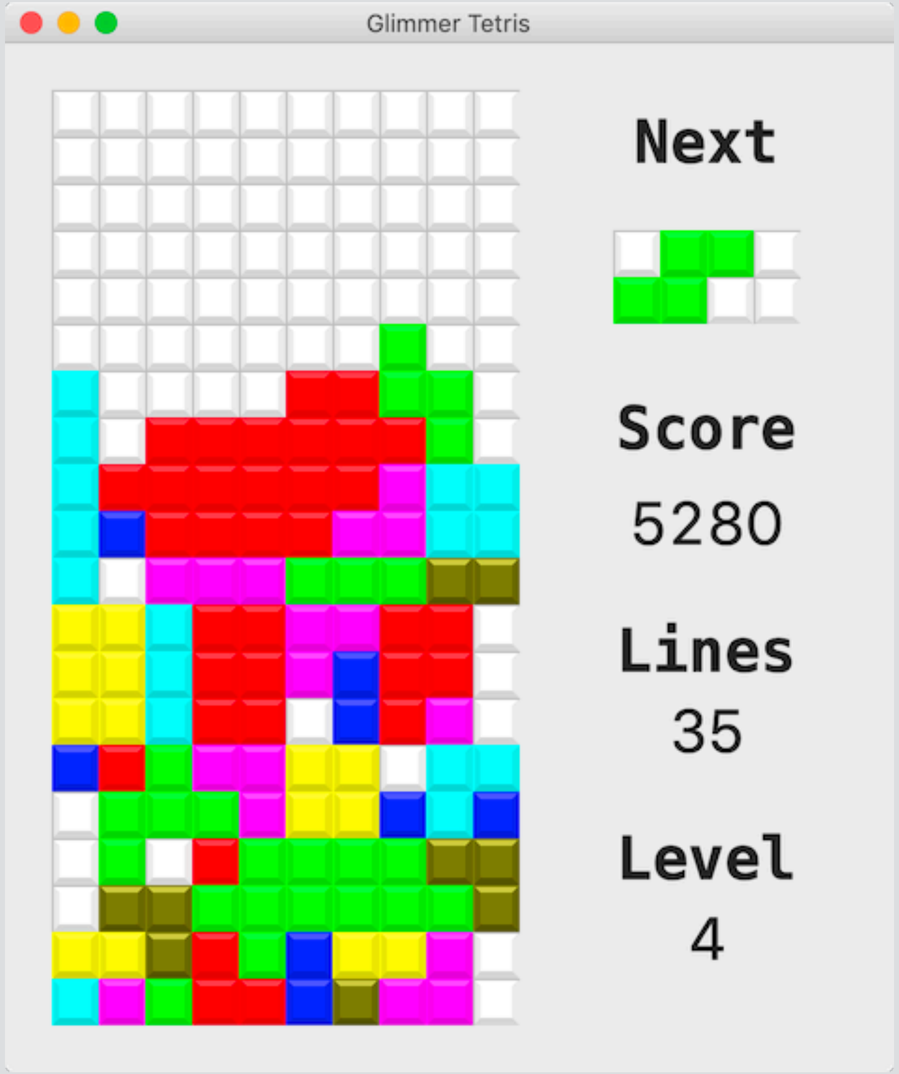
Shoes 4

```
Shoes.app width: 300, height: 200 do
  background lime..blue

  stack do
    para "Welcome to the world of Shoes!"
    button "Click me" do alert "Nice click!" end
    image "http://shoesrb.com/img/shoes-icon.png",
          margin_top: 20, margin_left: 10
  end
end
```



Glimmer

- Glimmer GUI DSL
  - Multiple backends (SWT, GTK, ...)
  - JRuby + SWT is the most mature
- JRuby makes cross-platform GUI much easier!
  - Works same everywhere
  - GUI libraries shipped with gem
- https://github.com/AndyObtiva/glimmer



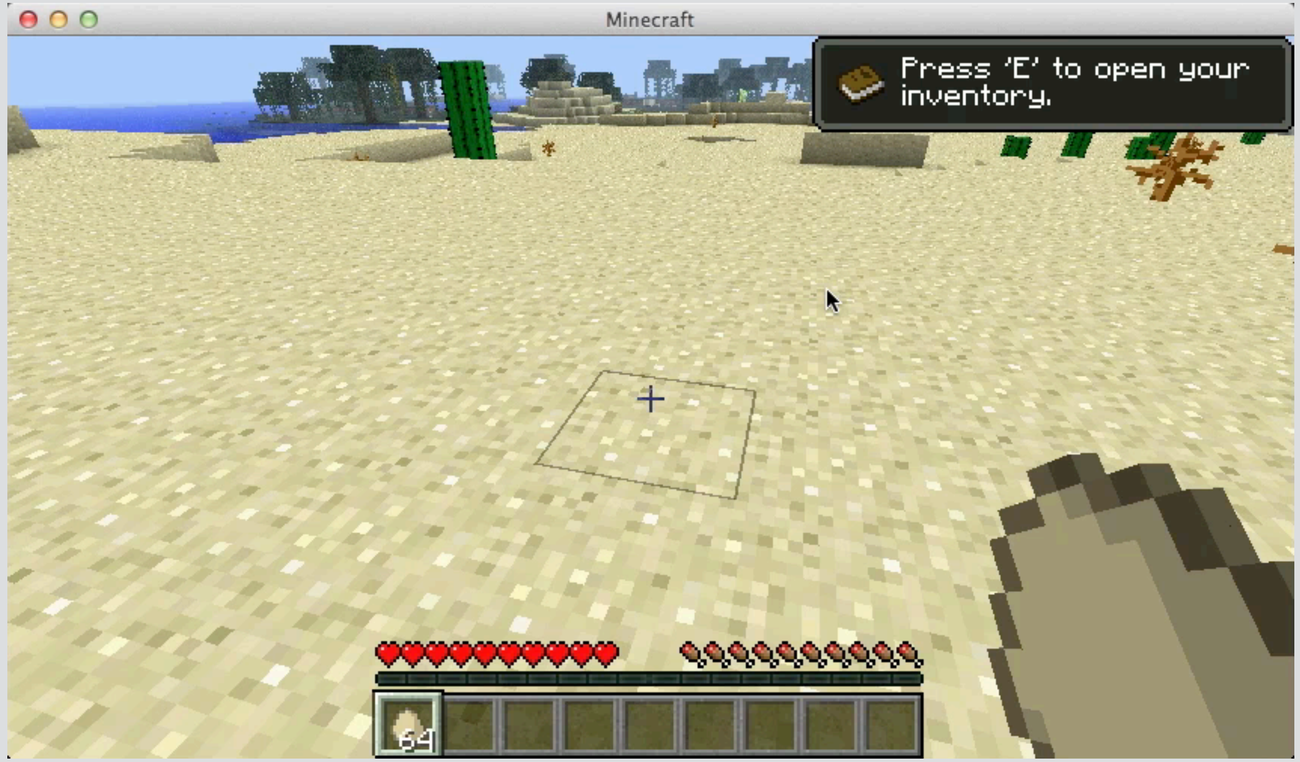
JavaFX

- Scene-based UI toolkit
- Cross-platform including mobile
- JRubyFX provides a Ruby wrapper
- FXML layout or widget scripting

```ruby
require 'jrubyfx'

class HelloJRubyFX < JRubyFX::Application

  def start(stage)
    with(stage, width: 300, height: 300, title: 'Hello JRubyFX') do
      layout_scene(:dark_blue) do
        group do
          rectangle(x: 10, y: 40, width: 50, height: 50, fill: :red) do
            translate_x = translateXProperty # note we must do this here
            timeline(cycle_count: :indefinite, auto_reverse: true) do
              animate translate_x, 0.sec => 1.sec, 0 => 200
            end.play
          end
        end
      end
      show
    end
  end
end

HelloJRubyFX.launch
```

Purugin for Minecraft

```
event(:player_egg_throw) do |e|
  e.hatching = true
  e.num_hatches = 120
  e.player.mesg "hatched"
end
```



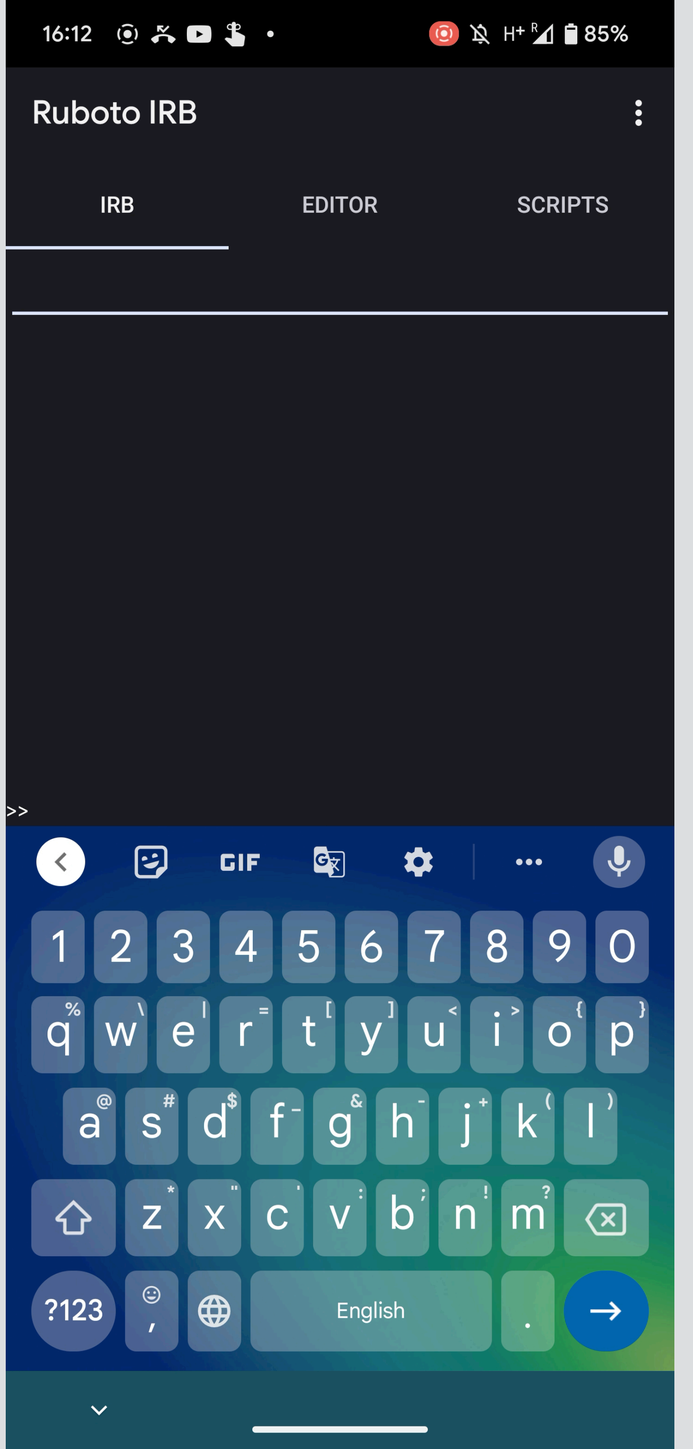
GUIs... like Android?

Ruboto: JRuby on Android

- ruboto.org, https://github.com/ruboto/ruboto
- Actively used for commercial projects today
- Build interface with GUI builder, wire it up with Ruby code
- Neglected a bit but being updated for JRuby 10 now!

Ruboto IRB

- IRB in JRuby on Android!
  - Plus an editor and script runner
- Not currently in the store, but we will republish soon!
- Search for "Ruboto Core" and "Ruboto IRB" APKs



JVM Features

JVM GC

- Many options to tune JVM GCs
  - Heap size: small or large?
  - Throughput: faster allocations or shorter pause times?
  - Working set: large in-memory or mostly new objects?
- Many options in standard OpenJDK
  - Serial, Parallel, G1, ZGC, Shenandoah

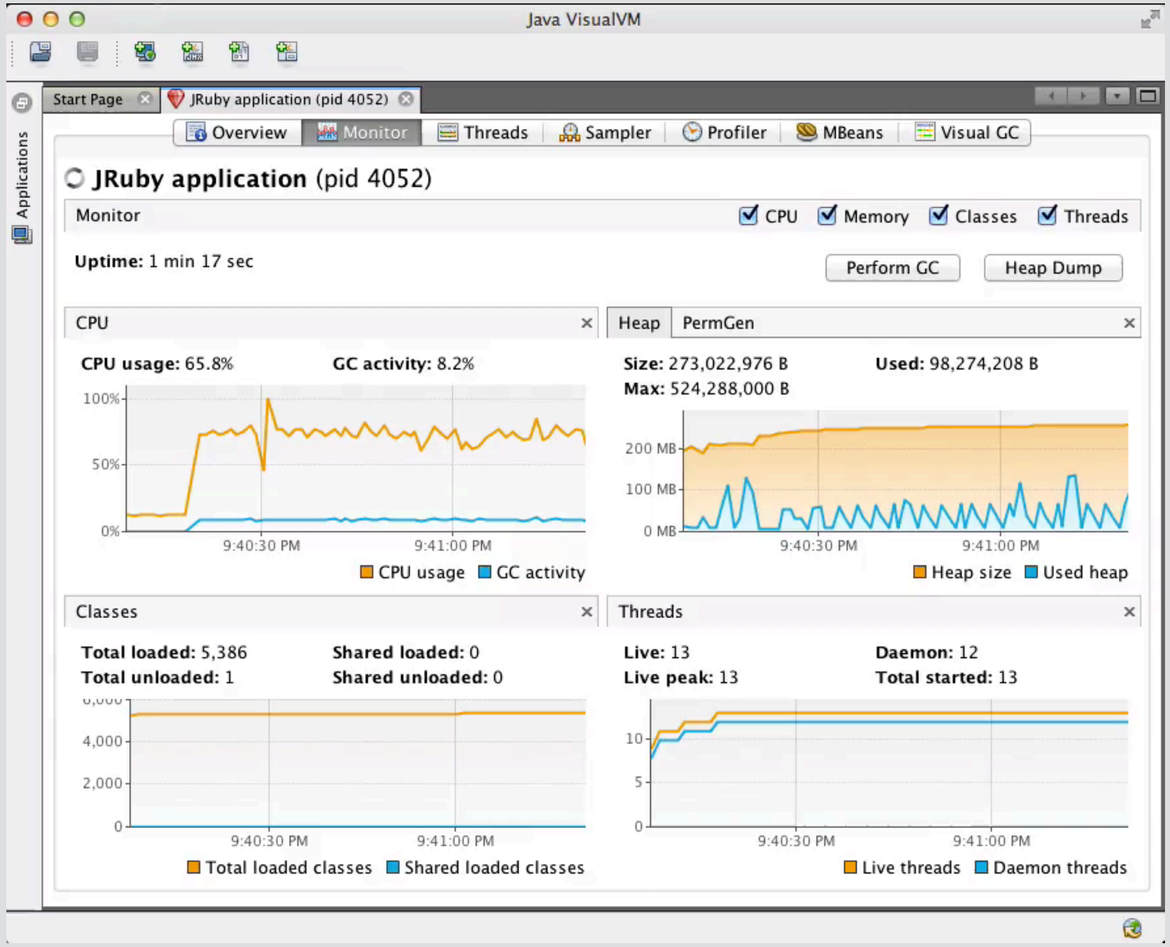
JVM JIT

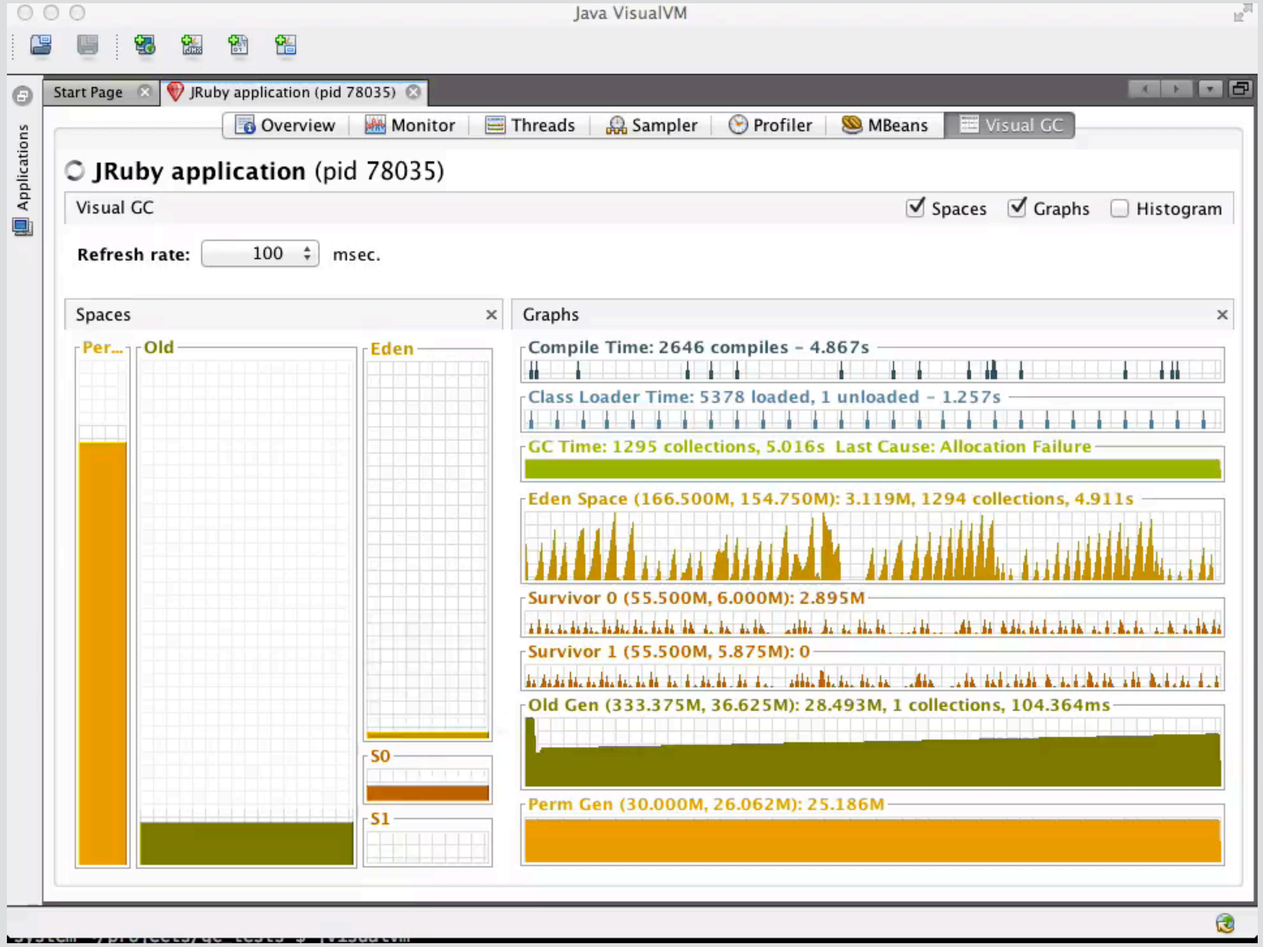
- HotSpot JIT is most widely deployed
  - C1 "client" JIT: fast, simple optimizations
  - C2 "server" JIT: needs profile data, heavy optimization
  - Both enabled with various "tiers" of JIT
- Graal JIT: newer, more aggressive optimizations
- OpenJ9 JIT (IBM JDK): offline AOT, JIT servers, CRIU

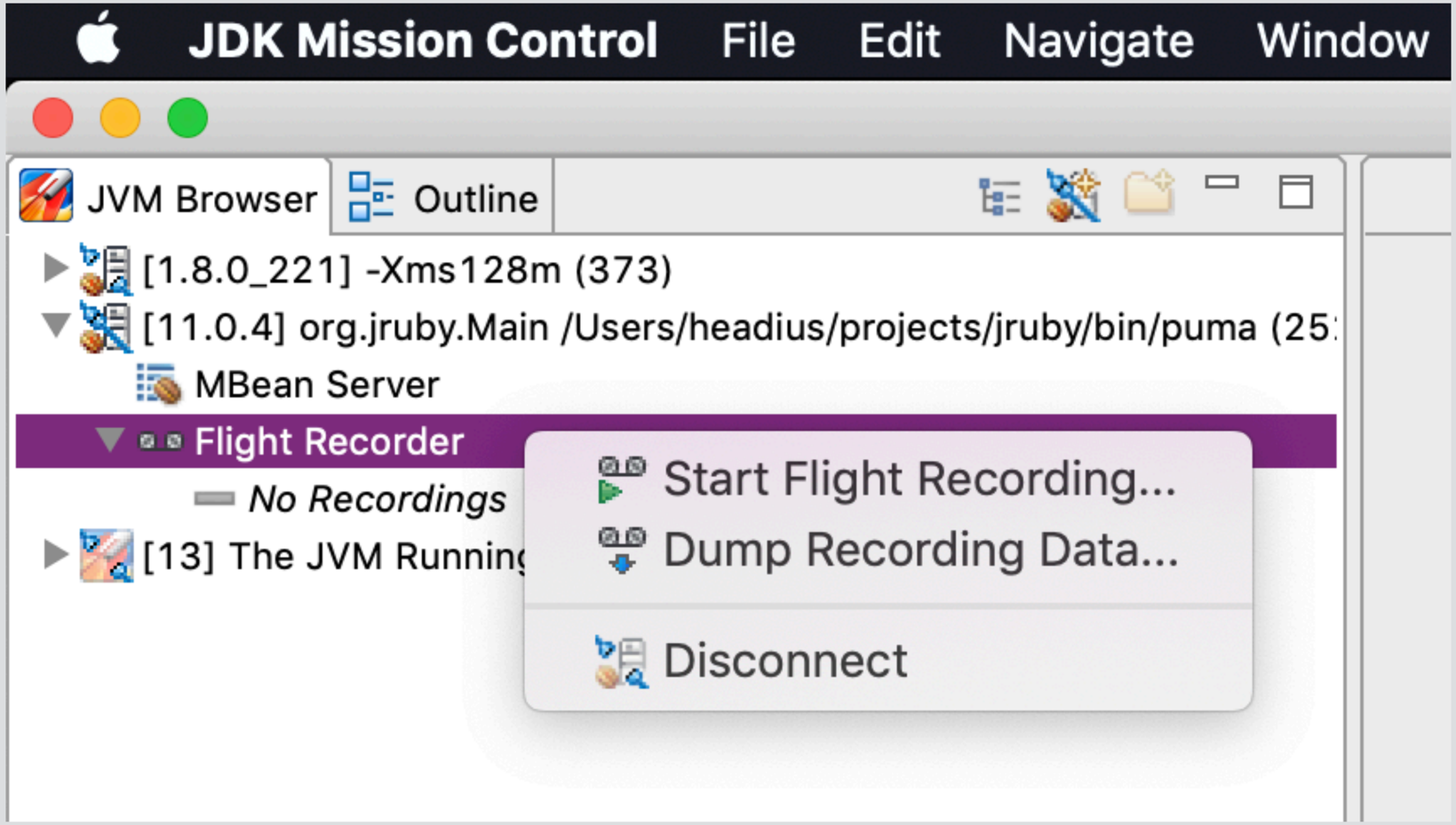
Monitoring and Profiling

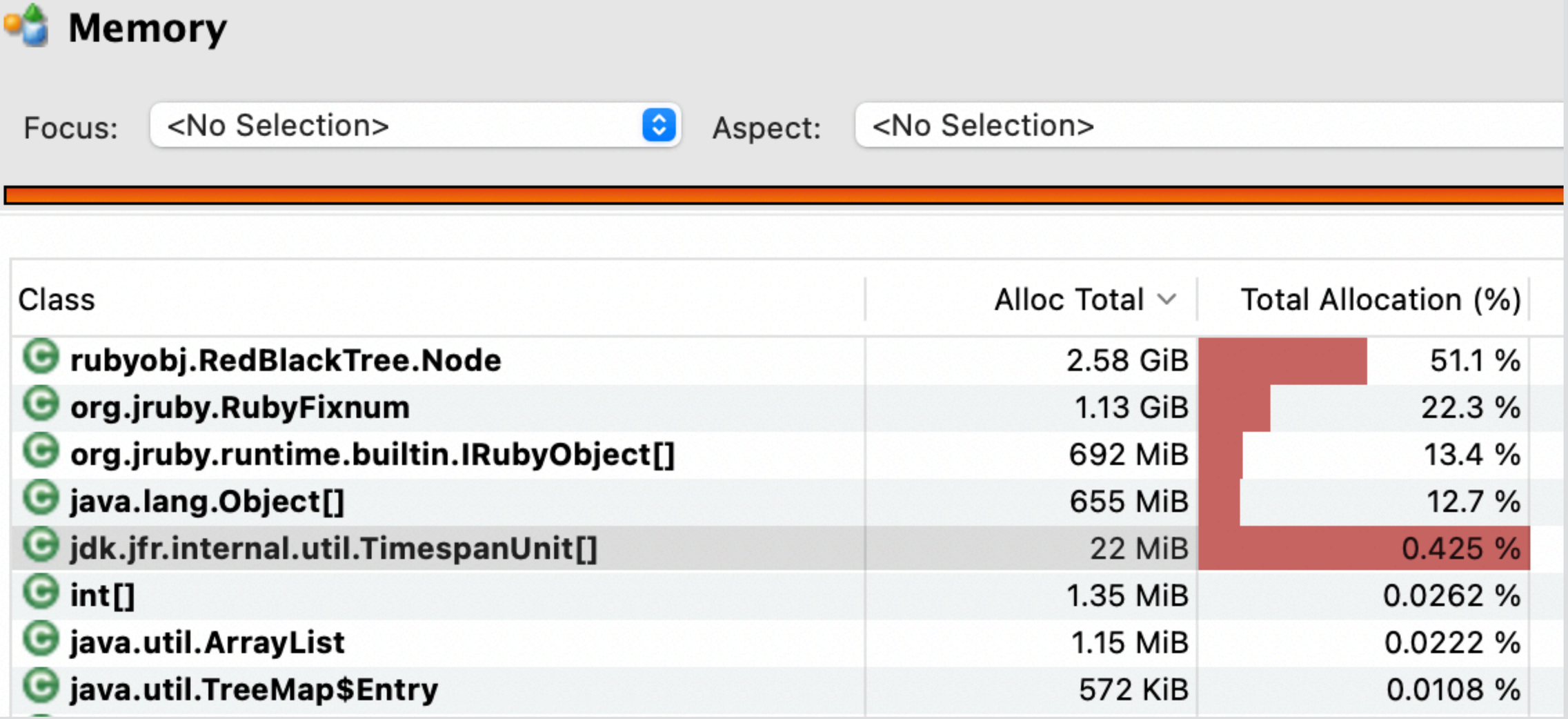
- VisualVM: GUI console for basic JVM monitoring
- JDK Flight Recorder: always-on monitoring with profiling options
  - Low-overhead 1% to 5%, built into OpenJDK
- JDK Mission Control: GUI client for Flight Recorder data
  - https://adoptium.net/jmc/

Visual VM


Java VisualVM
Start Page
JRuby application (pid 4052)
Overview
Monitor
Threads
Sampler
Profiler
MBeans
Visual GC
Applications
JRuby application (pid 4052)
Monitor
CPU
Memory
Classes
Threads
Uptime: 1 min 17 sec
Perform GC
Heap Dump
CPU
CPU usage: 65.8%
GC activity: 8.2%
100%
50%
0%
9:40:30 PM
9:41:00 PM
CPU usage
GC activity
Heap
PermGen
Size: 273,022,976 B
Used: 98,274,208 B
Max: 524,288,000 B
200 MB
100 MB
0 MB
Heap size
Used heap
Classes
Total loaded: 5,386
Shared loaded: 0
Total unloaded: 1
Shared unloaded: 0
4,000
2,000
0
Total loaded classes
Shared loaded classes
Threads
Live: 13
Daemon: 12
Live peak: 13
Total started: 13
10
5
Live threads
Daemon threads


Java VisualVM
Start Page
JRuby application (pid 78035)
Overview
Monitor
Threads
Sampler
Profiler
MBeans
Visual GC
Applications
JRuby application (pid 78035)
Visual GC
Spaces
Graphs
Histogram
Refresh rate:
100
msec.
Spaces
Per...
Old
Eden
S0
S1
Graphs
Compile Time: 2646 compiles - 4.867s
Class Loader Time: 5378 loaded, 1 unloaded - 1.257s
GC Time: 1295 collections, 5.016s Last Cause: Allocation Failure
Eden Space (166.500M, 154.750M): 3.119M, 1294 collections, 4.911s
Survivor 0 (55.500M, 6.000M): 2.895M
Survivor 1 (55.500M, 5.875M): 0
Old Gen (333.375M, 36.625M): 28.493M, 1 collections, 104.364ms
Perm Gen (30.000M, 26.062M): 25.186M


JDK Mission Control
File
Edit
Navigate
Window
JVM Browser
Outline
[1.8.0_221] -Xms128m (373)
[11.0.4] org.jruby.Main /Users/headius/projects/jruby/bin/puma (25
MBean Server
Flight Recorder
No Recordings
[13] The JVM Running
Start Flight Recording...
Dump Recording Data...
Disconnect


Memory
Focus:
<No Selection>
Aspect:
<No Selection>
Class
Alloc Total
Total Allocation (%)
rubyobj.RedBlackTree.Node
2.58 GiB
51.1 %
org.jruby.RubyFixnum
1.13 GiB
22.3 %
org.jruby.runtime.builtin.IRubyObject[]
692 MiB
13.4 %
java.lang.Object[]
655 MiB
12.7 %
jdk.jfr.internal.util.TimespanUnit[]
22 MiB
0.425 %
int[]
1.35 MiB
0.0262 %
java.util.ArrayList
1.15 MiB
0.0222 %
java.util.TreeMap$Entry
572 KiB
0.0108 %

Method Profiling

Focus: <No Selection> Aspect: <No Selection>

bench_red_black

Method	Count	Percentage
bench2018.bench_red_black.❤ def insert_helper #27(Thr	244	16.6 %
bench2018.bench_red_black.❤ def insert #14(ThreadCon	20	1.36 %
bench2018.bench_red_black.❤ def minimum #16(Thread(	19	1.29 %
bench2018.bench_red_black.❤ def maximum #17(Thread(	15	1.02 %
bench2018.bench_red_black.❤ def search #22(ThreadCo	9	0.611 %
bench2018.bench_red_black.❤ def delete_fixup #28(Thre	6	0.407 %
bench2018.bench_red_black.❤ def successor #18(Thread	2	0.136 %
bench2018.bench_red_black.❤ def delete #15(ThreadCon	1	0.0678 %
bench2018.bench_red_black.❤ def inorder_walk #20(Thr	1	0.0678 %

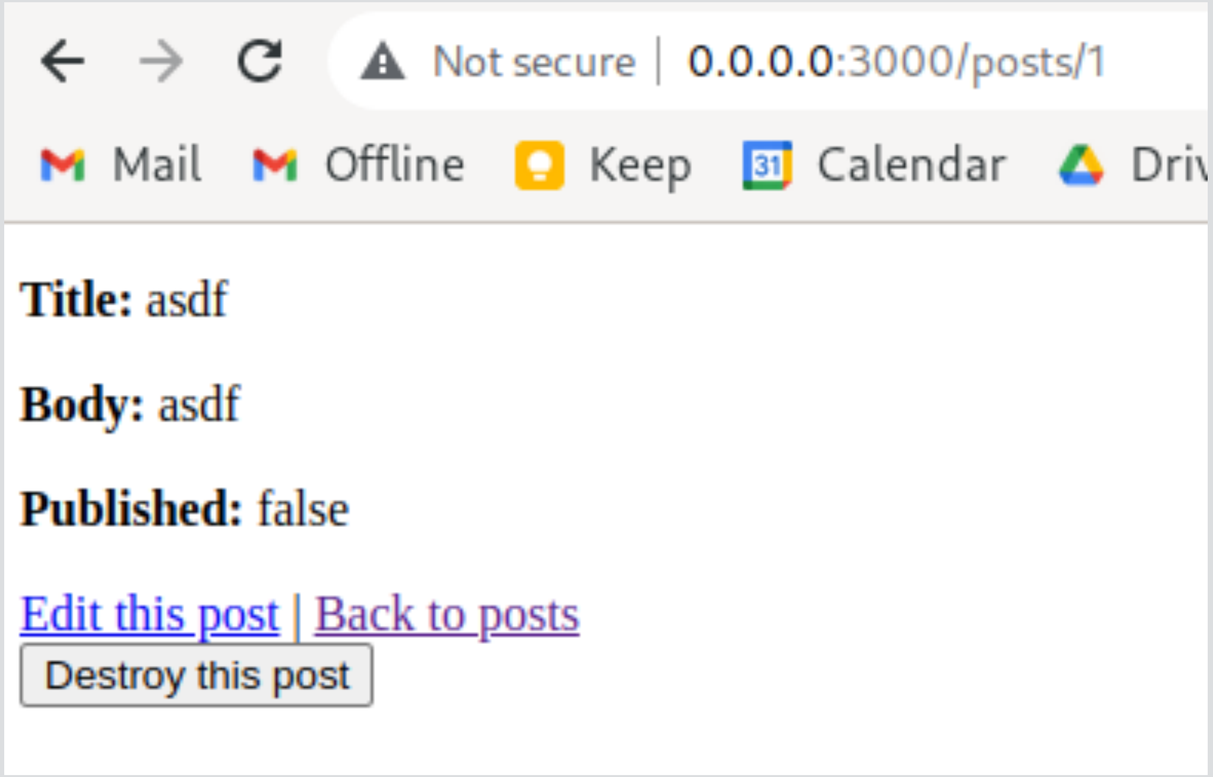
Performance and Scaling

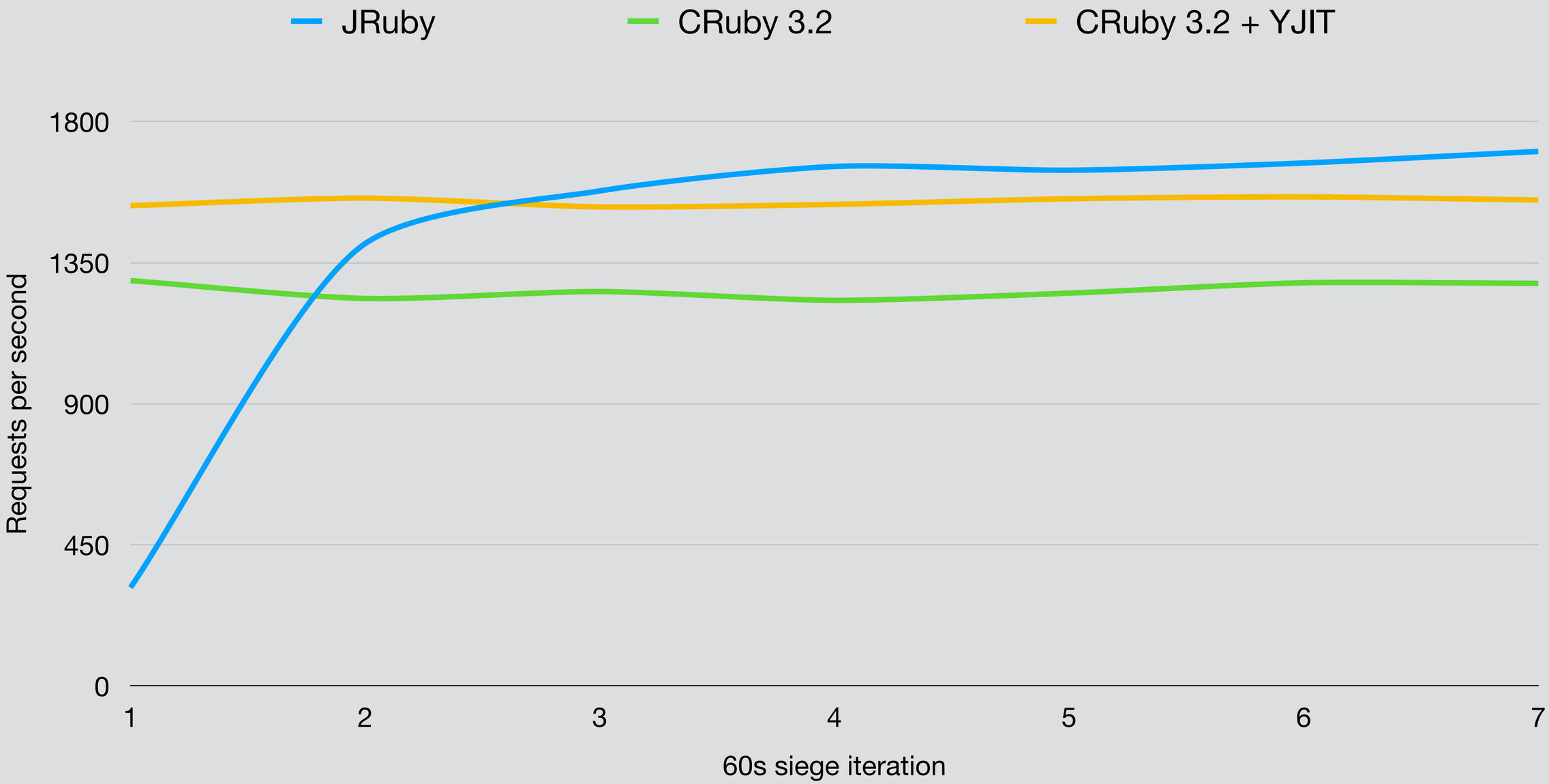
Scaling Applications

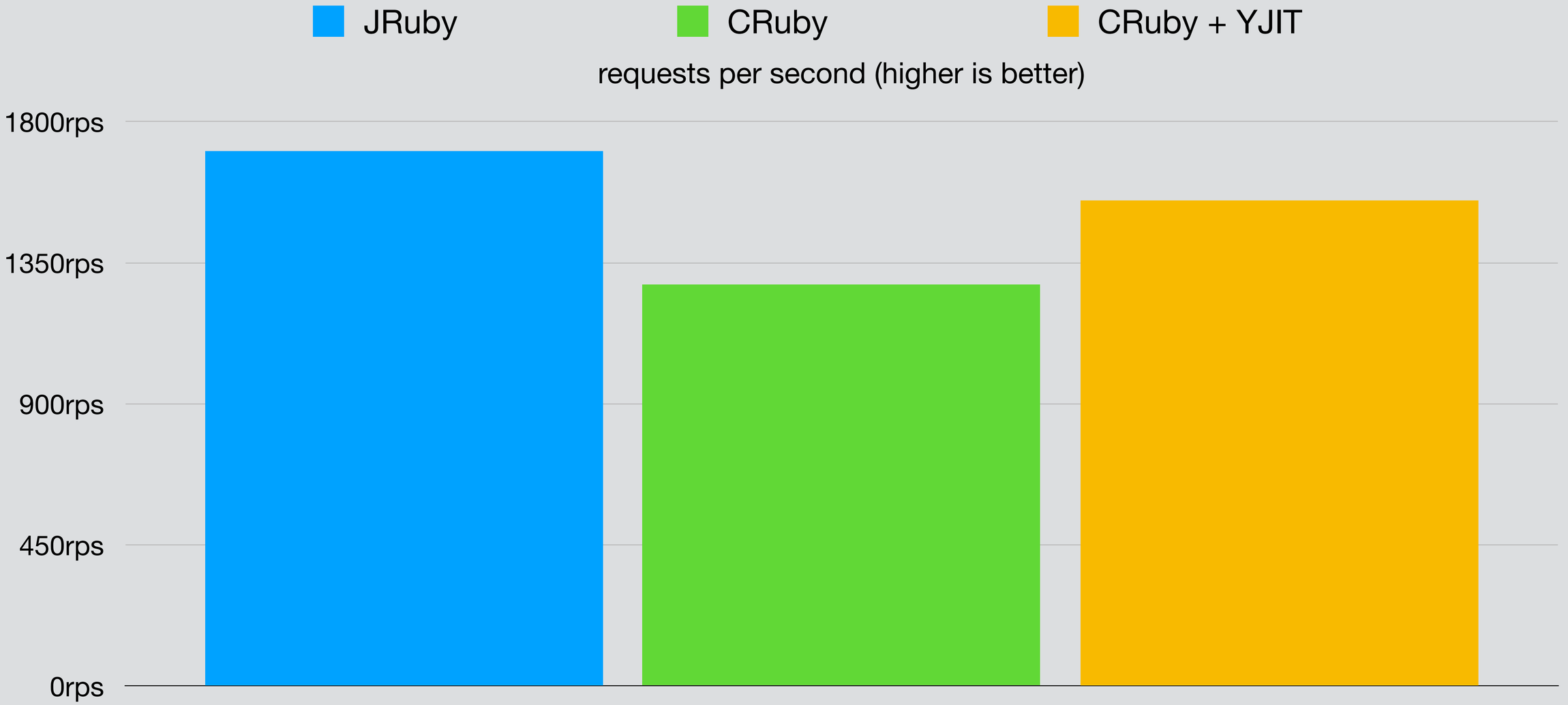
- Classic problem on CRuby/MRI
  - **No concurrent threads**, so we need worker processes
  - Processes **duplicate runtime state** and **waste resources**
- JRuby is a good solution
  - Multi-threaded **single process** runs your **entire site**
  - Single process with **leading-edge GC** uses resources better

Baseline Rails App

- Scaffolded "blog" application on PostgreSQL, Puma
- IBM VPC instance: 8 vCPU, 32GB
- CRuby 3.2, 16 workers
- JRuby 9.4: 16 threads
- Database, siege benchmark driver on same instance

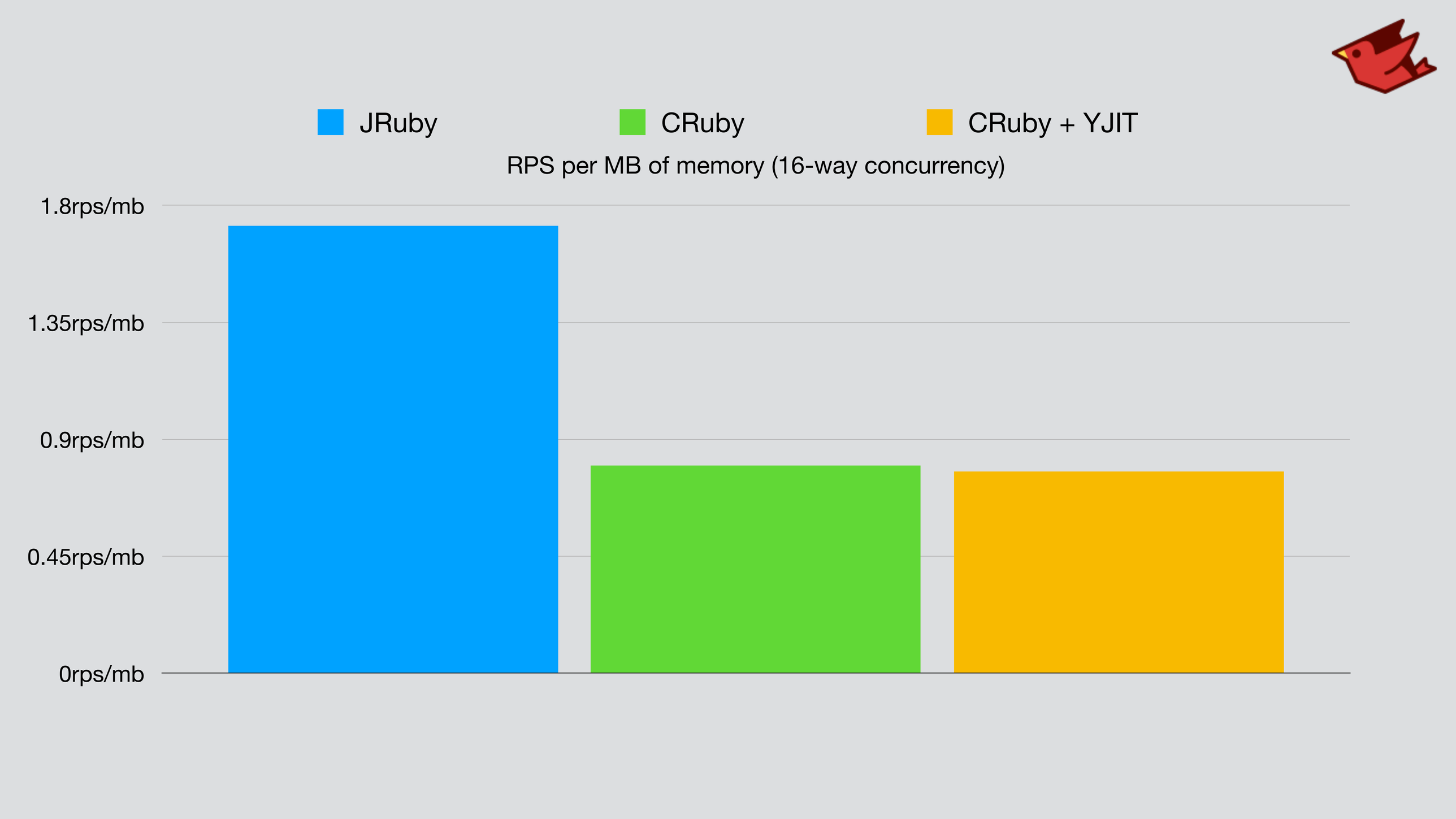


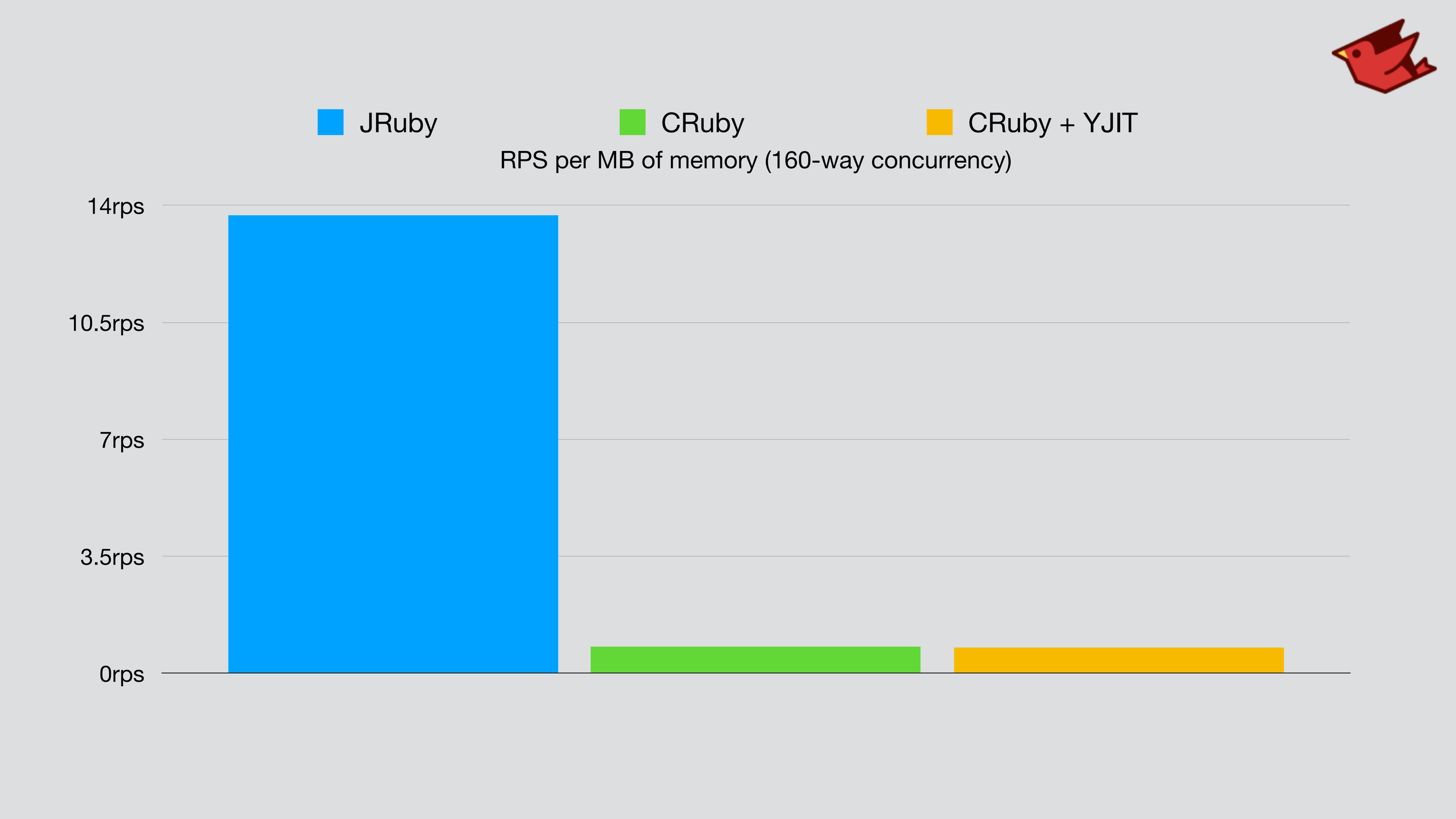

JRuby
CRuby 3.2
CRuby 3.2 + YJIT
1800
1350
900
450
0
Requests per second
1
2
3
4
5
6
7
60s siege iteration


JRuby
CRuby
CRuby + YJIT
requests per second (higher is better)
1800rps
1350rps
900rps
450rps
0rps

Memory

- JRuby: 3.4GB RSS
  - JRuby with 300MB heap: 955MB RSS
  - JRuby G1: 1.6G
- CRuby: 16x 103MB = 1.6GB
  - CRuby YJIT: 16x 125MB = 2GB


JRuby
CRuby
CRuby + YJIT
RPS per MB of memory (16-way concurrency)
1.8rps/mb
1.35rps/mb
0.9rps/mb
0.45rps/mb
0rps/mb


JRuby
CRuby
CRuby + YJIT
RPS per MB of memory (160-way concurrency)
14rps
10.5rps
7rps
3.5rps
0rps

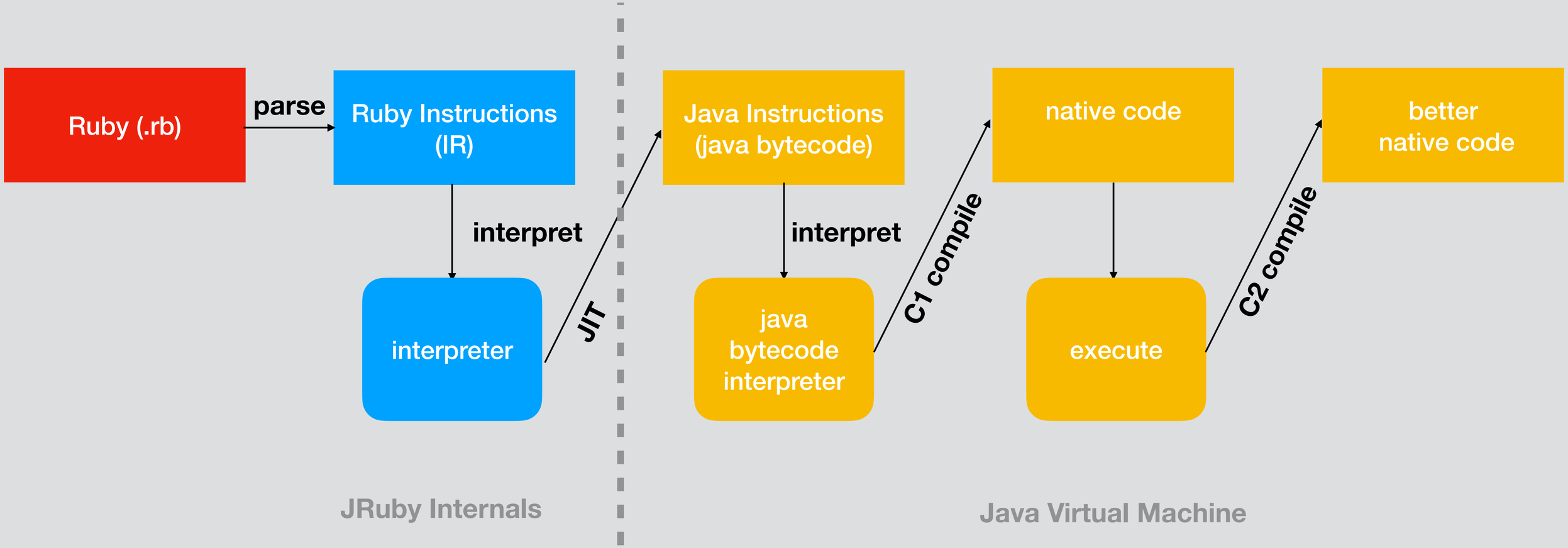
Optimizing JRuby 10

Optimizing Startup

Optimizing Startup

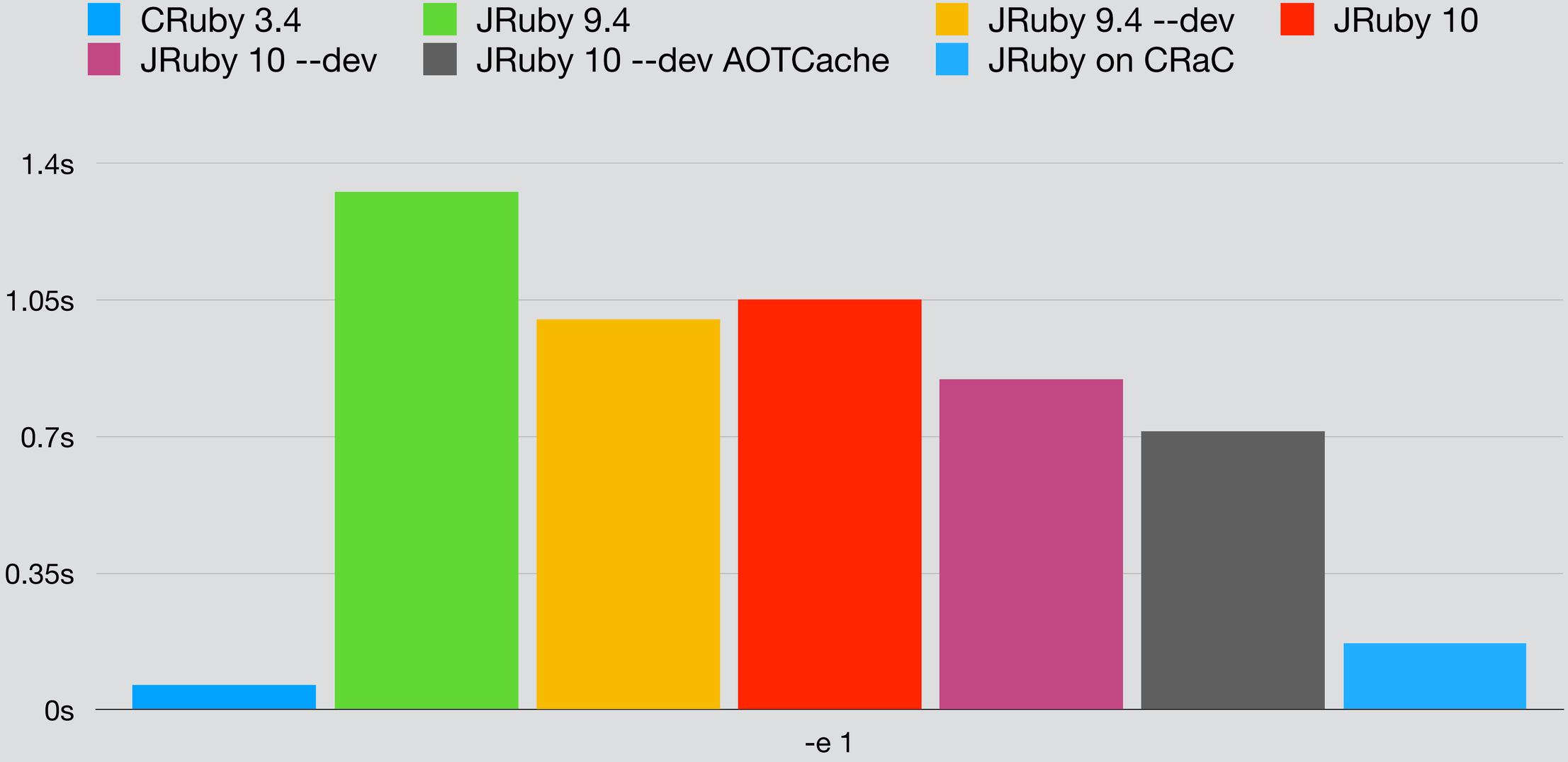
- First impressions are important!
- JRuby has always had much longer startup than CRuby
  - JVM must load, parse, optimize each time
  - Delaying parts of startup only helps "hello world"
- Newer JDKs shipping **improved startup features**

JRuby Compiler Pipeline



JVM Startup Projects

- Application Class Data Store (**AppCDS**)
  - Pre-cache classes, methods, code, metadata across runs
- Machine code caching (**AOTCache**)
  - AppCDS + execution profiles and native code
- Coordinated restore at checkpoint (**CRaC**)
  - Capture entire process state and restore later (repeatedly)

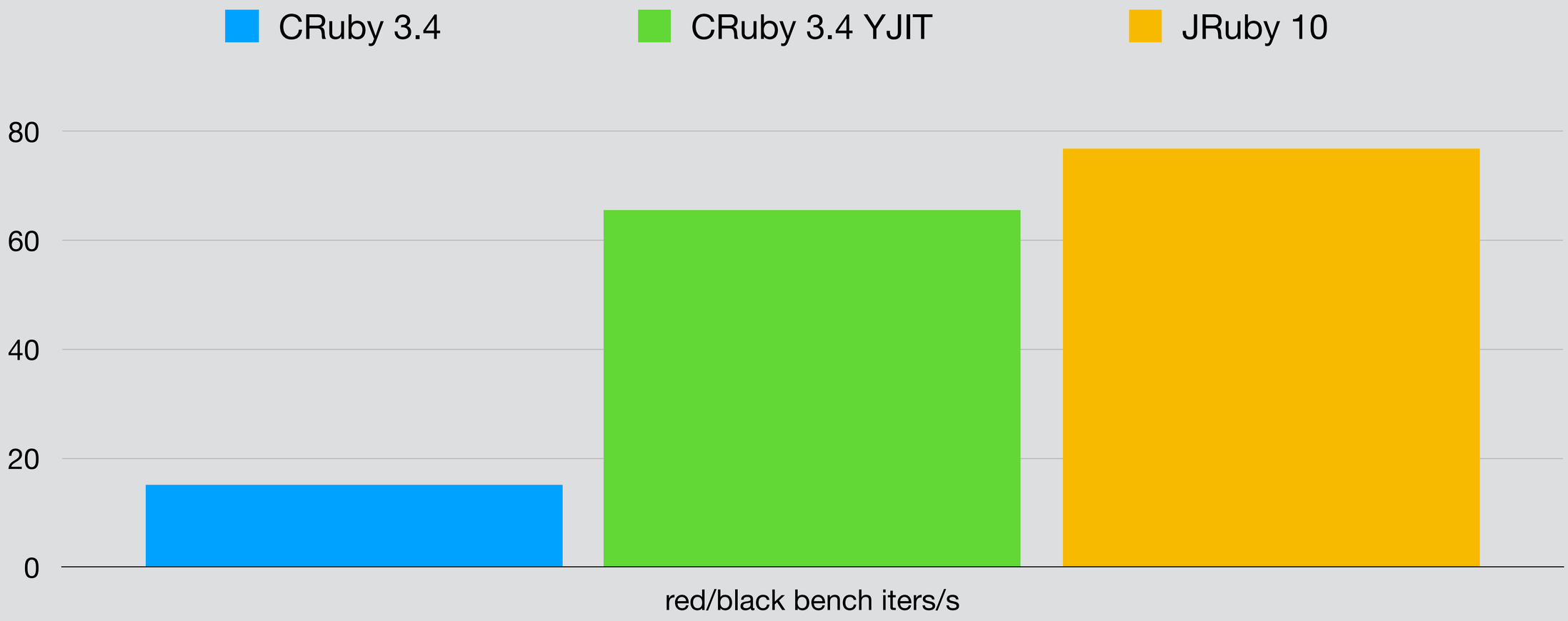

CRuby 3.4
JRuby 9.4
JRuby 9.4 --dev
JRuby 10
JRuby 10 --dev
JRuby 10 --dev AOTCache
JRuby on CRaC
1.4s
1.05s
0.7s
0.35s
0s
-e 1

Optimizing Performance

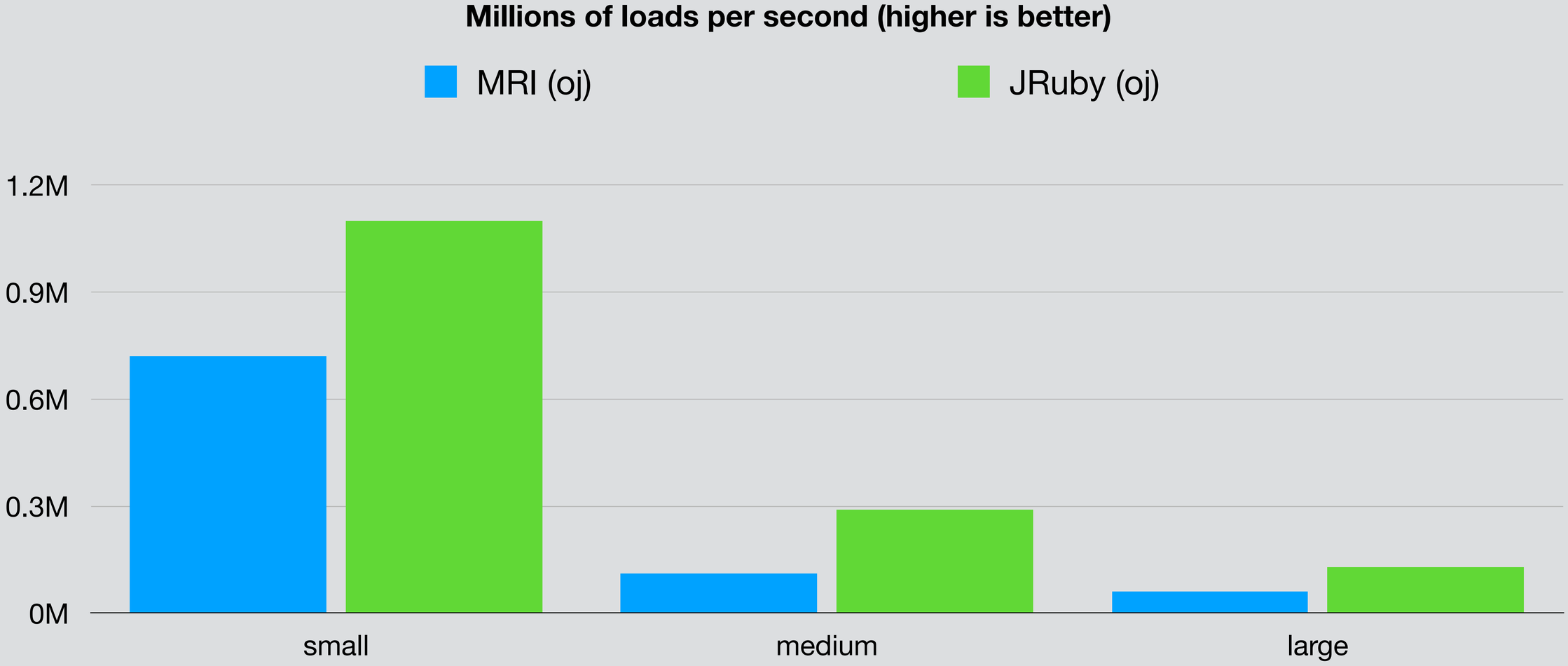
Optimizing Performance

- JRuby depends on JVM for most optimization
  - JVM has **world-class JIT compilers**, garbage collectors
  - Similar design to ZJIT, but with 30 years of work in it
- JRuby itself has an IR, basic block-based interpreter and JIT
  - Similar design to ZJIT starting in **JRuby 9.0** (2015)

Pure Ruby red/black tree



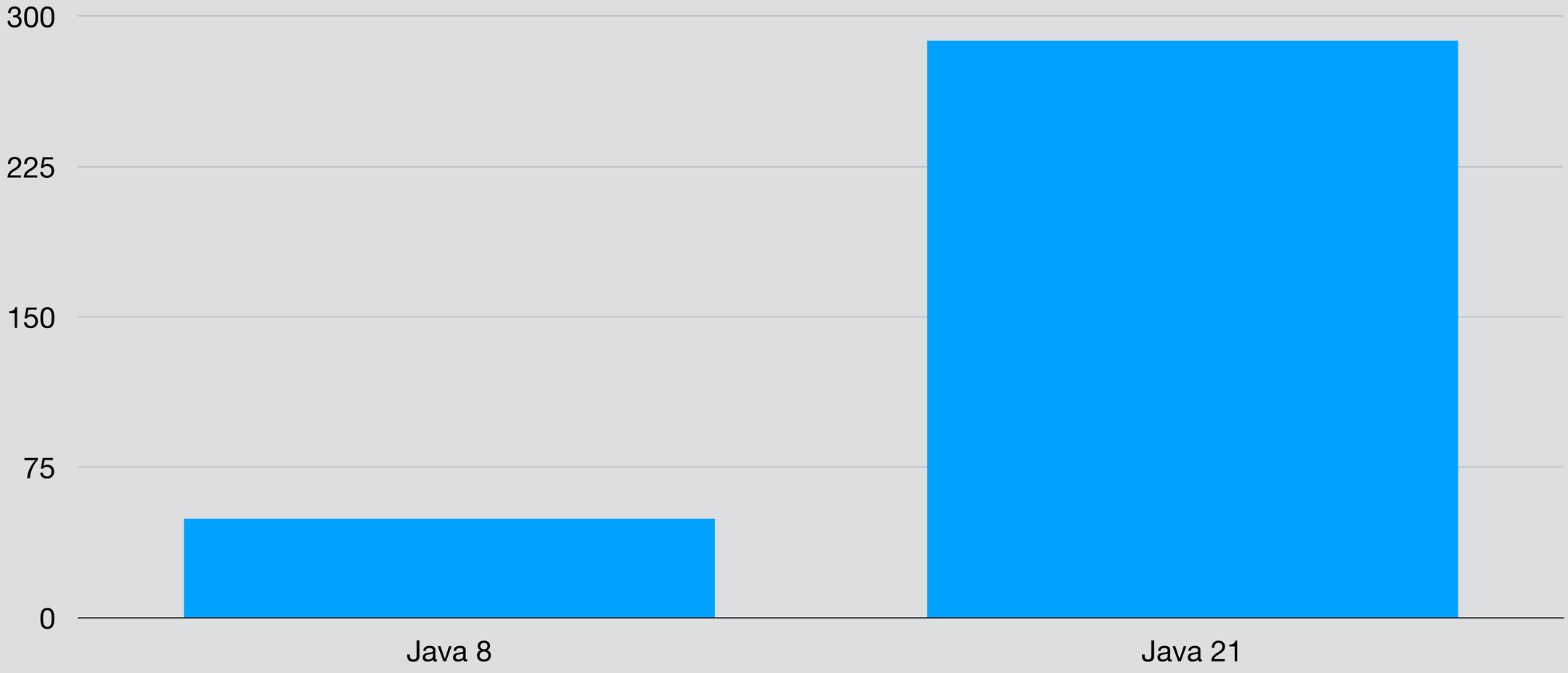
Better Extension Performance



Optimizing Fibers

- Fibers were not well-supported on older JVMs
  - Only abstraction was a native thread, too heavy for 1000s of fibers
- OpenJDK Project Loom adds fibers to JVM
  - "Virtual threads" are lightweight, but use same thread API
  - JRuby can match CRuby for count and perf of fibers

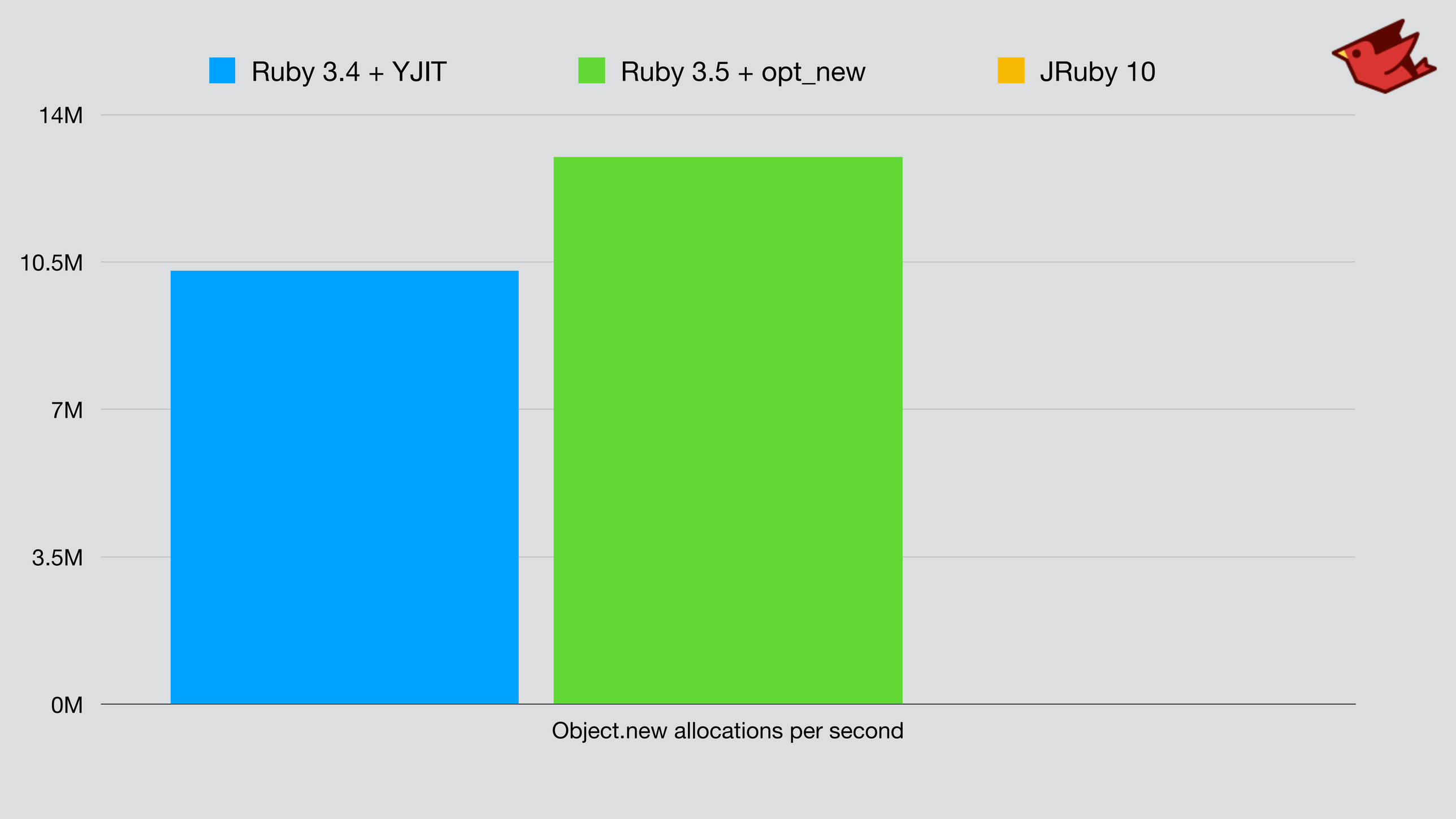
```
1000.times.map {
  Fiber.new { Fiber.yield }
}.each(&:resume).each(&:resume)
```

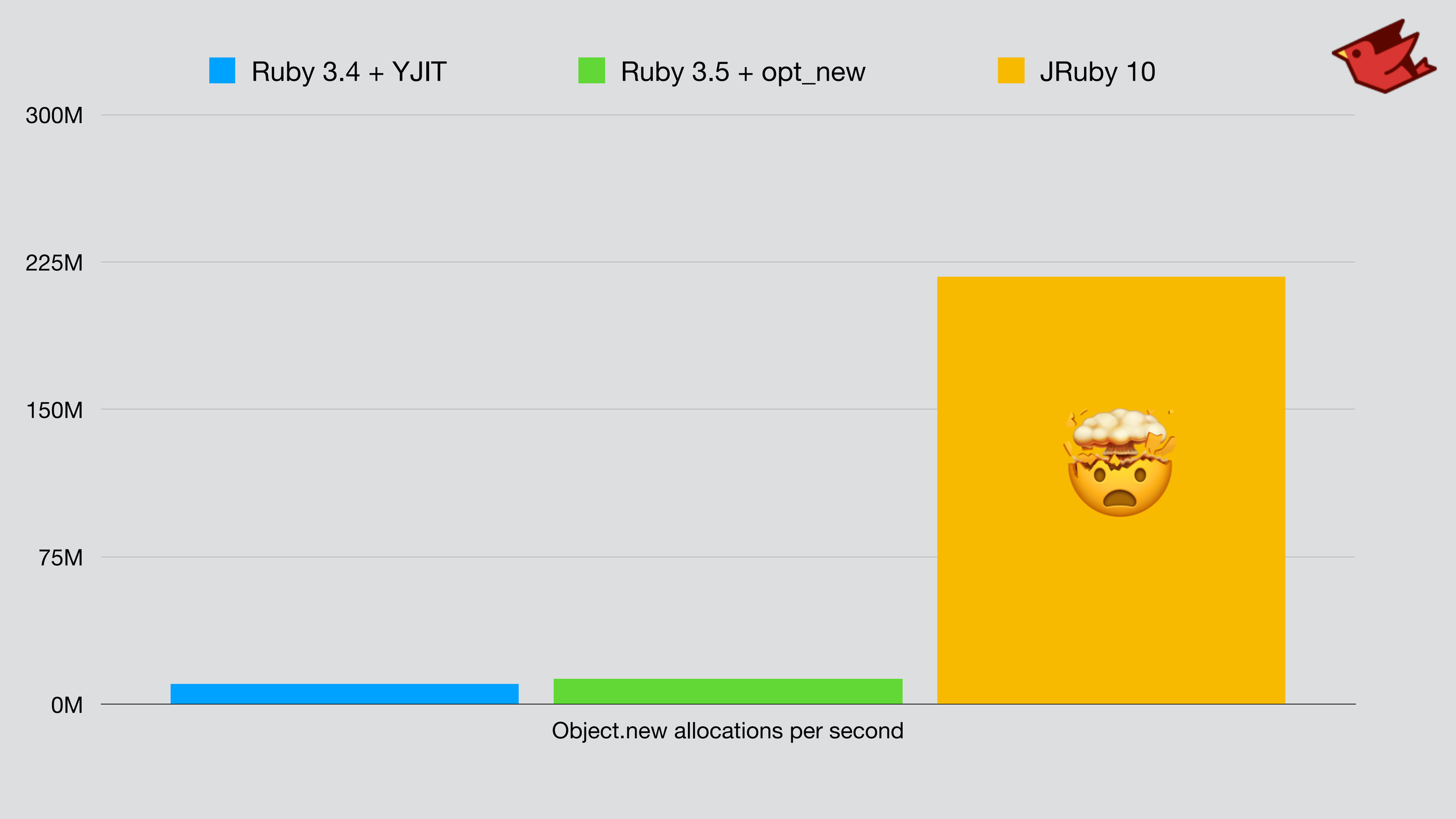

Create, resume, and finish 1000 fibers
300
225
150
75
0
Java 8
Java 21

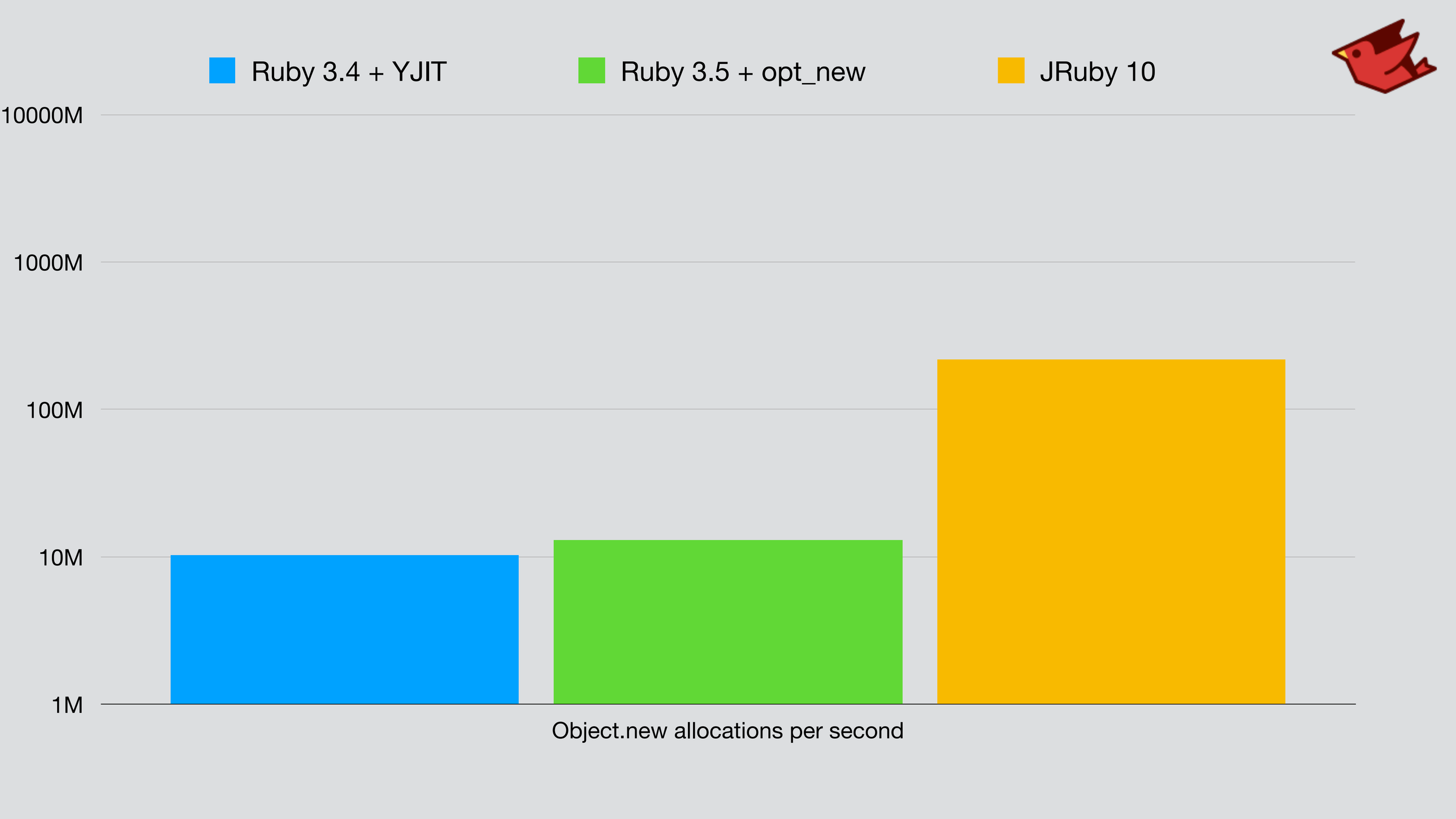
Class#new Optimization

- CRuby adding opt_new to optimize Class#new
  - Using C code to allocate + initialize hurts performance
- JRuby implemented this in 2016
  - If Class#new is default, allocate and call #initialize inline
  - Allocation and initialize both inline back to caller

Object.new


Ruby 3.4 + YJIT
Ruby 3.5 + opt_new
JRuby 10
14M
10.5M
7M
3.5M
0M
Object.new allocations per second

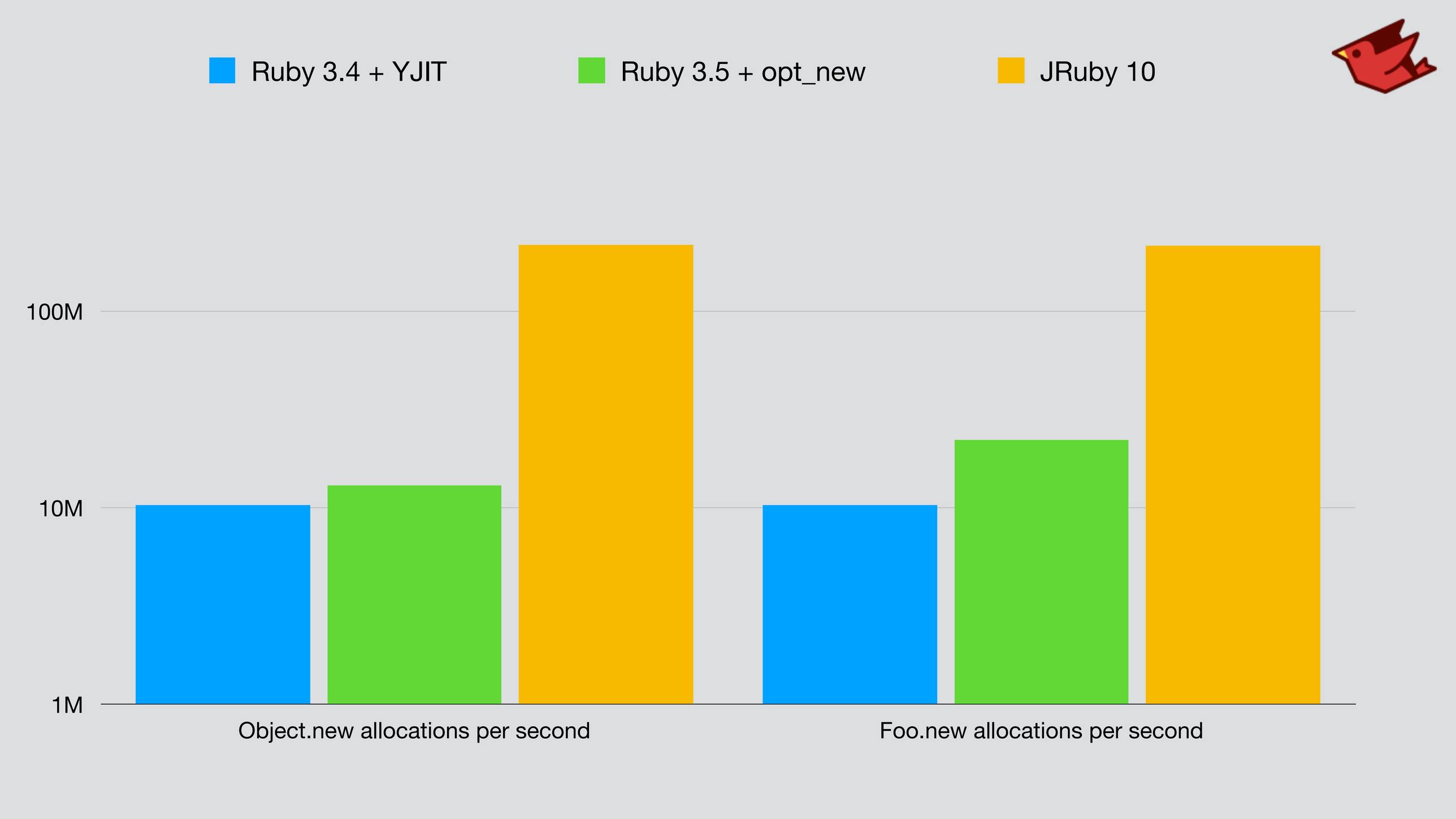

Ruby 3.4 + YJIT
Ruby 3.5 + opt_new
JRuby 10
300M
225M
150M
75M
0M
Object.new allocations per second


Ruby 3.4 + YJIT
Ruby 3.5 + opt_new
JRuby 10
10000M
1000M
100M
10M
1M
Object.new allocations per second

Allocation Profile

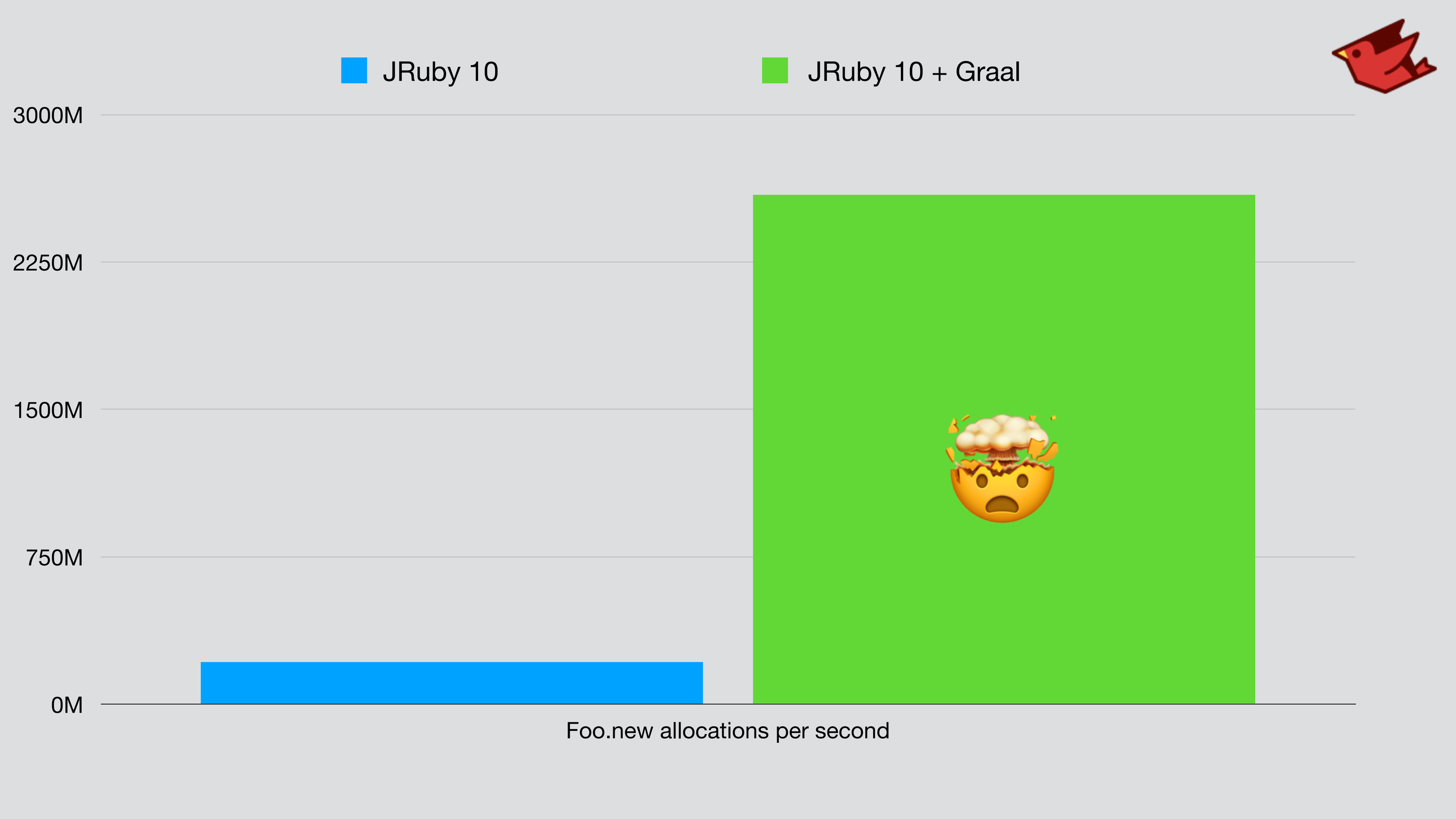
Class	Alloc Total	Total Allocation (%)
rubyobj.Foo	43.7 GiB	98.7 %
org.jruby.RubyFixnum	545 MiB	1.2 %
int[]	19.5 MiB	0.0431 %
java.util.ArrayList	4.75 MiB	0.0105 %

```
class Foo
  def initialize = nil
end
```


Ruby 3.4 + YJIT
Ruby 3.5 + opt_new
JRuby 10
100M
10M
1M
Object.new allocations per second
Foo.new allocations per second

Alternative JITs

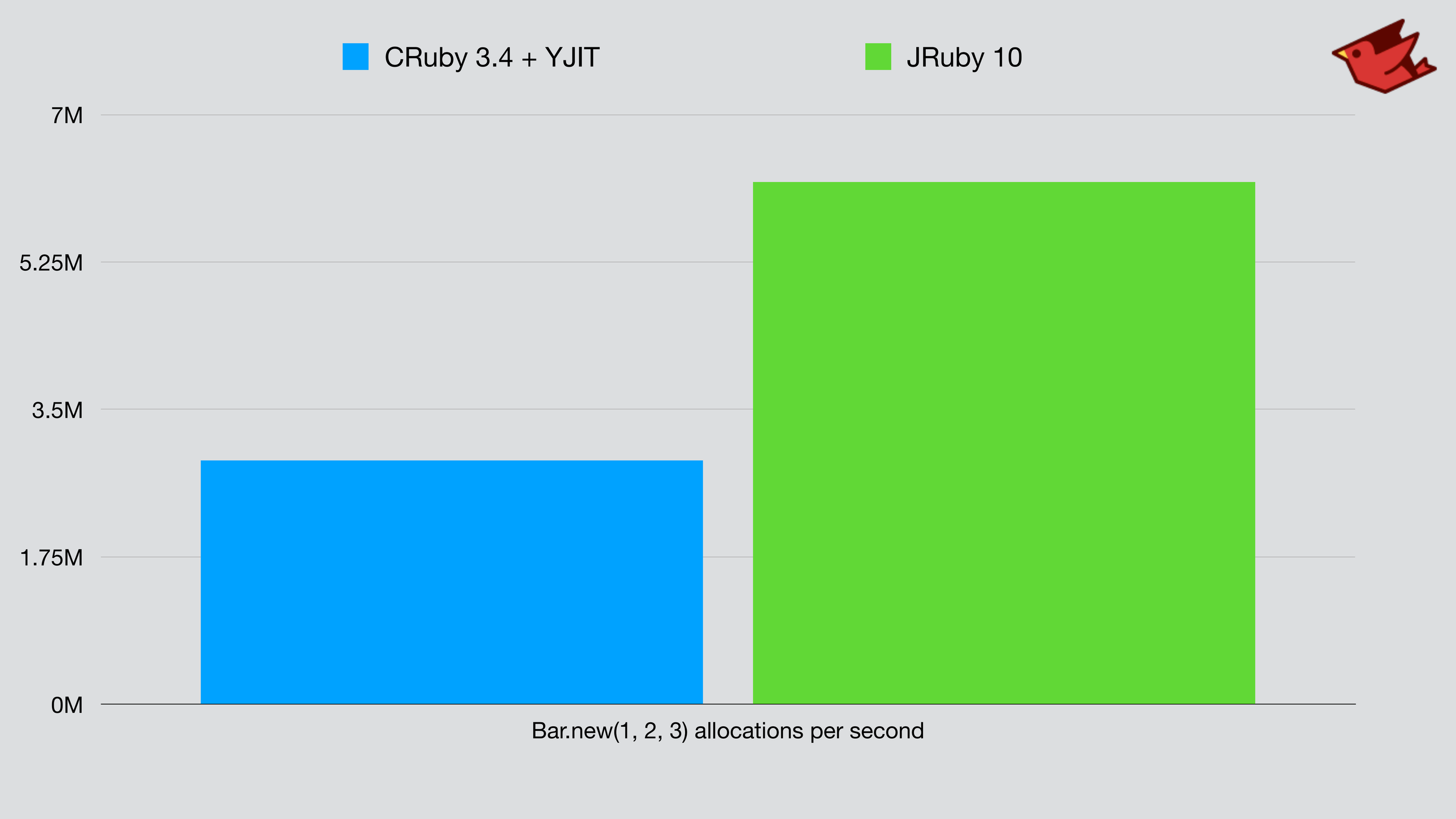
- HotSpot JIT
  - Standard JIT compiler in OpenJDK
  - Best balance of performance, reliability
- Graal JIT
  - Part of GraalVM project
  - Advanced optimizations, sometimes unpredictable perf


JRuby 10
JRuby 10 + Graal
3000M
2250M
1500M
750M
0M
Foo.new allocations per second

Data

- New immutable struct type in Ruby 3.2
- Compact storage
- Special #new method to allocate and initialize together

```
Bar = Data.new(:a, :b, :c)
Bar.new(1, 2, 3)
```


CRuby 3.4 + YJIT
JRuby 10
7M
5.25M
3.5M
1.75M
0M
Bar.new(1, 2, 3) allocations per second

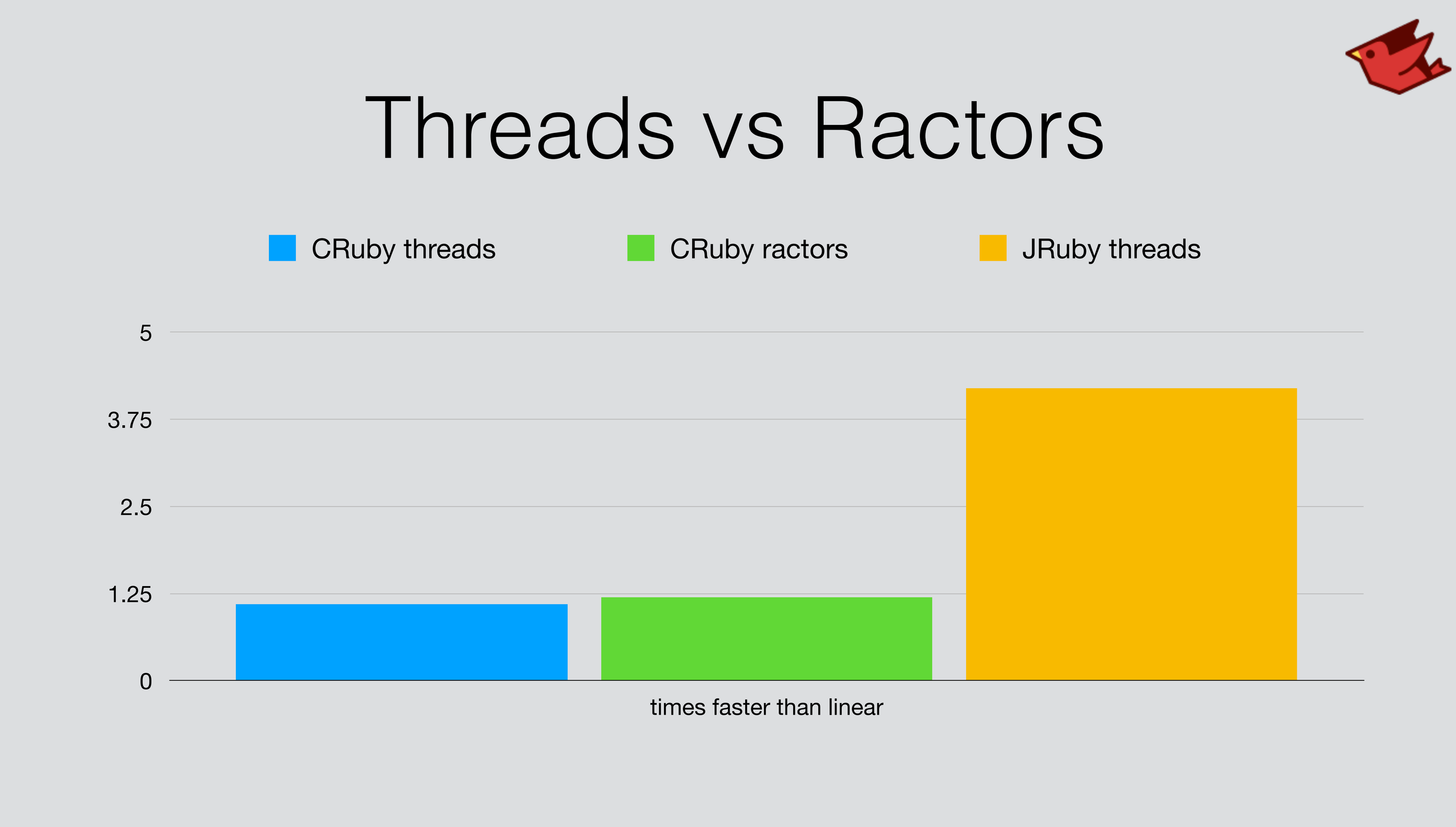
Optimizing Concurrency

Threads or Ractors?

- Ractors are designed to bring concurrency to Ruby
  - You must write Ractor-friendly code
  - High overhead crossing Ractor boundary
- Threads in JRuby are already 100% concurrent
  - You must right Thread-friendly code
  - But zero overhead due to shared object
- Thanks to Maciej Mensfeld for the benchmark

```
if THREADS
  queues = CONCURRENT.times.map { Queue.new }
  threads = CONCURRENT.times.map do |i|
    Thread.new do
      queue = queues[i]
      data_chunk = queue.pop
      data_chunk.each { |item| JSON.parse(item) }
    end
  end
end
```

```
result = Benchmark.measure do
  if THREADS
    CONCURRENT.times do |i|
      queues[i] << data[i]
    end
    threads.each(&:join)
  else
    # Linear without any threads
    data.each do |piece|
      piece.each { JSON.parse(_1) }
    end
  end
end
```


Threads vs Ractors
CRuby threads
CRuby ractors
JRuby threads
5
3.75
2.5
1.25
0
times faster than linear

JRuby Future

Supporting JRuby

- I work for you!
  - Helping Ruby users with JRuby
  - Keep making JRuby better
- Sponsorships from users
  - Every little bit helps!
- JRuby and Ruby support
  - Migration, updates, performance work

JRuby 10.x

- Many more optimizations coming!
  - Now that we are 3.4 compatible, I can work on fun stuff
  - If you find something slower than CRuby, tell me!
- More JDK features
  - AOTCache for startup, Panama for fast native calls
  - Upgrade JVM, your JRuby code runs faster!

Thank You!

- Charles Oliver Nutter
- @headius
  - @mastodon.social
  - .bsky.social
- headius@headius.com
- Help me keep working on JRuby!